Computer Science

Projective Geometry and Photometry for Object Detection and Delineation

Jefferey A. Shufelt

July 29, 1996
CMU-CS-96-164

Carnegie Mellon

Projective Geometry and Photometry for Object Detection and Delineation

Jefferey A. Shufelt

July 29, 1996

CMU-CS-96-164

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

The author was supported by an Augmentation Award for Science and Engineering Research Training sponsored by the Army Research Office, Department of the Army, under Contract DAAH04-93-G-0092. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office, the Department of the Army, or the U.S. government.

Keywords: Computer vision, cartographic feature extraction, object detection, aerial image interpretation, building detection, monocular image analysis, photogrammetry, projective geometry, vanishing points, model-based recognition

Carnegie Mellon

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

Projective Geometry and Photometry for Object Detection and Delineation

JEFFEREY A. SHUFELT

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

[illegible] THESIS COMMITTEE CHAIR — 8/20/96 DATE

[illegible] DEPARTMENT HEAD — 8/21/96 DATE

APPROVED:

[illegible] DEAN — 8-26-96 DATE

Thesis Abstract

Computer vision systems have traditionally performed most effectively in constrained situations, where limitations on object shape or scene structure permit reliable image analysis. For example, in model-based recognition systems, the existence of 3D models of objects of interest allows the application of geometric constraints to limit the search for interpretations of low-level image information.

Many problem domains, however, do not have explicit constraints on object shape or scene content. In aerial image analysis, man-made structures take on a wide variety of shapes and sizes. Existing techniques for these domains obtain only partial solutions by the use of simplifying assumptions about imaging geometry, illumination conditions, and object shape. Few of these techniques attempt to model perspective or photometric effects, which can be powerful constraints for object detection and delineation.

The central hypothesis of this work, that rigorous modeling of the image acquisition process leads to improved detection and delineation of basic volumetric forms for object recognition, leads to the formulation of a set of principles for object detection and delineation. In accordance with these principles, a fully automated monocular image analysis system, PIVOT, was developed for the task domain of cartographic building extraction from aerial imagery, using original techniques for vanishing point detection, intermediate feature generation, hypothesis generation, and building model verification.

A quantitative comparative evaluation methodology for object detection and delineation is presented in this work, using unbiased image space and object space performance metrics on large datasets of imagery. Using this methodology, PIVOT was compared to three existing building extraction systems on 83 test images covering a wide variety of geographical areas, object complexities, and viewing angles. This analysis demonstrates PIVOT's improved performance from highly oblique viewpoints and on complex manmade structures, establishing the utility of rigorous camera modeling for object detection and delineation tasks, and in particular its importance for the automated population of spatial databases with cartographically accurate three-dimensional models.

Acknowledgements

I met Dave McKeown for the first time eight years ago, during my junior year as an undergraduate at CMU. I remember very little about that meeting, but I do remember that he seemed to know quite a bit about Division I collegiate basketball. It turned out, of course, that he faked a great deal of it, as I found out when I started running the lab's NCAA pool, and his Final Four picks fell by the wayside year after year.

Fortunately, his performance as an advisor and a close friend was in no way correlated with his ability to predict the outcome of basketball games. Much of my research style and philosophy, as well as my background in automated cartography and computer vision, can be traced directly to Dave, and his constant stream of ideas, insights, and suggestions have been invaluable throughout my career. Special thanks go to Dave for his collaboration, encouragement and support.

Thanks are due to Eric Grimson, Katsushi Ikeuchi, and Chris McGlone, for serving on the thesis committee; their comments and suggestions significantly improved the development and exposition of this work. Particular thanks go to Chris, who introduced me to the science of photogrammetry and the mathematics behind it, and who collaborated with me on earlier work which served as the seed for this thesis.

A major strength of the Computer Science department is its people. I have enjoyed working with Tom Mitchell, Avrim Blum, Bernd Bruegge, and particularly Hans Berliner on a variety of projects. Thanks also go to Sharon Burks, for just being Sharon Burks. There has been no shortage of entertainment in the office thanks to Denis Dancanet, Peter Jansen, See-Kiong Ng, Conrad Poelman, and Jiri Sgall. Thanks are also due to Gary Aranyos, Rene Brigode, and Mark Puskar for keeping everything running smoothly.

Special thanks go to the past and present members of the Digital Mapping Laboratory for providing an exciting and interesting working environment, as well as their participation in extended foosball research sessions. It has been great fun. In particular, the research in this thesis has been influenced by numerous discussions with Steven Cochran, Stephen Ford, Steve Gifford, Wilson Harvey, Yuan Hsieh, Bruce Irvin, Frederic Perlant, Mike Polis, Michel Roux, and Aviad Zlotnick. Yuan Hsieh deserves special mention here, as his efforts on the SITECITY semi-automated site modeling system were done in parallel with this work, resulting in many fruitful conversations about computer vision, geometric modeling, and performance evaluation. Thanks also go to Ted Bulwinkle and Nate Segerlind for their work on SITECITY ground truth measurements and edge extraction, respectively.

Keys Botzum, Brad Cline, Chris Hoover, Andrew Nelson, and Karen O'Kane all deserve thanks for their friendship and support throughout this work. Laura Ashley and family ensured that I met my minimum dietary requirements for grilled meat. Denis Dancanet and Yuan Hsieh merit special recognition for their roles in daily no-holds-barred discussions over lunch. Steering the conversational straits between *la chupacabra* and the King's Gambit requires a deft hand at the oars.

I am particularly grateful for the love and encouragement of my family, David, Connee, Rob, and Katie. They have never failed to support me in all of my endeavors, and this work has been no exception.

Finally, I thank Stacey Jacobs for love, patience, and understanding.

Table of Contents

Chapter 1
Introduction

Three-dimensional object recognition has been a major focus of computer vision research for many years, due to its potential for automating tasks in such diverse problem domains as cartographic database compilation, industrial inspection and assembly, and autonomous navigation. In the cartographic domain, a successful recognition system would greatly reduce the effort needed to assemble a digital map product. An effective industrial recognition system would permit the automation of tedious manual inspection and assembly tasks. The ability to efficiently and reliably recognize objects gives a robot the ability to intelligently navigate and interact with its surroundings.

It is well understood that a general solution to the recognition problem must address the combinatorial problems of matching sensory data to plausible interpretations of that data. To date, 3D recognition systems have traditionally performed most effectively in constrained situations, where limitations on object shape or scene structure permit reliable image analysis. Model-based recognition systems exploit the geometry of object models, often in controlled imaging environments, to constrain the search for interpretations of low-level image information. These constraints allow systems to focus their efforts on small portions of the hypothesis space that are likely to contain plausible interpretations.

Unfortunately, many problem domains do not have strong constraints on object shape or scene content. In the domain of aerial image analysis, for example, manmade structures take on a wide variety of shapes and sizes; the use of a model-based approach here leads to a formidable model library compilation problem. Given a lack of explicit information about the objects of interest, existing approaches in this domain have relied on simplifying assumptions about imaging geometry, illumination conditions, and object shape, producing only partial solutions to the general problem. This naturally leads to several related questions.

In the absence of explicit object models:

- what does it mean to "find an object?"
- what constraints can we employ to recognize objects without undertaking an exhaustive search of hypothesis space?
- how can we verify that we have found an object?

The first question is answered by relying on a set of recent computational and psychological theories which suggest that objects are represented as compositions of a few basic volumetric forms. This answer reduces the problem of finding an object to two subproblems; finding instances of the forms that comprise the objects, and connecting these forms when appropriate.

The answer to the remaining questions forms the central argument of this thesis:

> *Basic volumetric forms for object recognition can be detected, delineated, and verified with cues derived from rigorous modeling of the image acquisition process.*

The validity of the central argument of this thesis is supported by the development and extensive experimental evaluation of a computer vision system for object detection and delineation in complex real-world images. This evaluation leads to two key results:

- Rigorous camera modeling enables powerful geometric and photometric constraints for object detection and delineation. Even in unconstrained environments, the perspective geometry induced by the imaging process provides strong cues for constraining interpretations of image data. Given the position of the camera and the light source, shadow geometry and illumination analysis can be utilized to verify the presence of 3D structure. These constraints play a significant role in reducing the search space for feature extraction algorithms.
- These constraints are sufficient for the extraction of basic volumetric forms from complex imagery, leading to flexible and robust modeling of generic objects from complex scenes under a wide range of viewpoints. By detecting, delineating, and combining simple models which have fixed shapes but variable extents, a wide range of object types can be modeled with camera-derived geometric constraints, without the need for explicit models for all objects in a scene.

The organization of this work reflects the processing flow of the computer vision system used to validate the central argument of the thesis. PIVOT (**P**erspective **I**nterpretation of **V**anishing points for **O**bjects in **T**hree dimensions) is an implementation of the key elements of the geometric and photometric constraints, in conjunction with primitive object models, that represent the main contributions of this work.

Chapter 2 discusses the object detection and delineation problem in detail, and presents six guiding principles for the development of a robust high-performance object detection and delineation system. These principles are motivated by an analysis of the strengths and weaknesses of existing techniques, and supported by examples derived from real aerial photography. The principles presented in this chapter serve as motivation for the design of PIVOT.

Chapter 3 presents *primitives*, the basic modeling units of PIVOT, and *vanishing points*, the basic geometric cues used to detect primitives. This chapter makes two novel contributions: the use of geometric constraints derived from the primitives to guide a search for vanishing points, and error models which incorporate uncertainty in image edges to provide robust vanishing point solutions.

Chapter 4 describes a data-driven approach for object hypothesis generation, moving from raw image edge data through increasingly detailed and complex intermediate representations to three dimensional instantiations of primitives. The techniques in this chapter make heavy use of the geometric information described in Chapter 3 to constrain the potentially combinatorially explosive search for primitives to a manageable one.

Chapter 5 addresses the verification problem; given an set of instantiated primitives, select those which best model the structure of the scene. An important aspect of the verification process is the use of 3D modeling of shadow effects and qualitative surface illumination constraints, enabled by the use of a photogrammetric camera model.

The focus of Chapter 6 is performance analysis. Given the lack of serious quantitative analysis in previous work, as highlighted in Section 1.1, a key aspect of the thesis is this chapter's thorough comparative analysis of PIVOT's performance with other monocular building extraction systems on a wide variety of complex aerial scenes, using unbiased metrics to characterize system performance in detection and delineation.

Finally, Chapter 7 summarizes the contributions of this thesis, and discusses their impact on the object detection and delineation problem in light of the principles proposed in Chapter 2. Appendix A contains mathematical preliminaries which the reader may find useful for reference, and Appendix B presents a compilation of experimental results which form the basis for the performance analysis discussion in Chapter 6.

In the remainder of this introduction, a survey of the literature on 3D object detection and delineation is presented. We briefly consider model-based approaches, and then we turn to the domain of aerial image analysis, where much work has been done on the problem of generic object detection and delineation, using a variety of techniques and assumptions. This survey leads naturally to a consideration of the approach advocated in this work.

1.1. A survey of previous research

Before discussing the research on 3D object analysis, it is useful to define a few basic terms. The term *recognition* has been used haphazardly in the literature; in some cases it has referred to the ability to attach semantic or functional interpretations to 3D object models, while in other cases it has referred to the ability to perform a coarse segmentation of an image to distinguish regions by photometric, geometric, textural, or spectral properties. This ambiguous terminology can lead to confusion; some systems reported to perform general object recognition in fact only produce coarse scene descriptions with no semantic attachment.

In this work, the following definitions will be used:

- *detection*: the classification of image pixels into regions corresponding to objects of interest.
- *delineation*: the placement of region boundaries in image space which project precisely to physical boundaries in object space.
- *recognition*: the attachment of semantic information to objects in a scene.

These definitions distinguish between the act of finding an object (detection), precisely marking its boundary (delineation), and determining what it is (recognition). Note that the definitions do not form a strict hierarchy: recognition does not always imply delineation, since some image understanding

prior research	acquisition, shape, and photometric assumptions	system analysis; result type	quantitative evaluation	coordinate space
Nagao, Matsuyama (1980) *region growing, multi-spectral classification*	nadir acquisition geometry, good edge and shadow contrast	detection, recognition; blob	none	image space
Tavakoli, Rosenfeld (1982) *intensity-based edge grouping*	nadir acquisition geometry, locally straight and sharp edges, buildings brighter than background, anti-parallel sides	detection; unconnected boundary segments	none	image space
Conners, Trivedi, Harlow (1984) *texture classification*	nadir acquisition geometry, good contrast in textured areas	detection, recognition; coarse grid classification	evaluation of coarse grid classification	image space
McKeown, Denlinger (1984) *region growing and template matching*	nadir acquisition geometry, good building edge contrast	detection, recognition; blob	none	image space
Herman, Kanade (1986) *line-corner junction analysis* *and truth maintenance for* *geometric reasoning*	good building edge contrast, buildings are trihedral polyhedra, all lines and surfaces lie in horizontal or vertical planes, surfaces with parallel lines have same height	detection, delineation; 3D wireframes	none	image space, 3D by verticals, stereo
Harwood, Chang, Davis (1987) *region growing and heuristic merging*	nadir acquisition geometry, good building edge contrast	detection, recognition; blob	none	image space
Nicolin, Gabler (1987) *region-based segmentation,* *perceptual grouping of building hypotheses*	nadir acquisition geometry, good building edge contrast	detection, recognition; blob	none	image space

Table 1-1: Early aerial image building detection approaches

algorithms can assign meaning to image regions whose boundaries do not strictly conform to object boundaries. Similarly, delineation does not strictly imply detection: it is possible to perform boundary fitting on regions without any notion of a specific object of interest.

Much research on 3D object recognition has been model-based; comprehensive surveys of this work can be found in [6, 9, 16, 31]. Model-based approaches to object recognition are predicated on the assumption that recognition implies full knowledge of object structure. However, humans are capable of detecting and delineating objects never seen before, and even in instances where the object semantics are unknown, humans can still classify objects qualitatively by shape, pending knowledge of object function [7].

Another concern with model-based approaches is the necessity of an adequately populated object library, particularly in domains where the number of distinct objects is large. Manual acquisition of a sufficient number of models for aerial images would be a formidable task; automated model acquisition in this domain would require unconstrained object detection and delineation ability, which is exactly the problem being considered.

A variety of approaches have been suggested for generic object recognition. A survey of object recognition strategies [97] discusses a variety of recognition algorithms, some of which employ more generic models of object structure that do not enforce specific shapes, and are more accurately classified as detection and delineation algorithms. These methods typically exploit regularities in image data, such

prior research	acquisition, shape, and photometric assumptions	system analysis; result type	quantitative evaluation	coordinate space
Huertas, Nevatia (1988) *line-corner analysis, shadow-based corner labeling*	nadir acquisition geometry, buildings composed of sequences of right-angled corners, buildings brighter than shadows	detection, delineation; closed polygonal boundaries	none	image space, 3d by shadow
Irvin, McKeown (1989) *box extrapolation from shadow regions, verification and grouping of boxes based on shadows*	nadir acquisition geometry, approximate buildings by rectangular boxes, buildings brighter than shadows	detection, delineation; four-sided boxes	none	image space, 3d by shadow
Mohan, Nevatia (1989) *edge-based perceptual grouping, network-based constraint satisfaction*	nadir acquisition geometry, approximate buildings by combinations of rectangular boxes	detection, delineation; composite rectangular boundaries	none	image space, 3d by stereo
Liow, Pavlidis (1990) *2 methods: shadow edge detection followed by region growing; region growing followed by shadow verification*	nadir acquisition geometry, buildings brighter than shadows, good contrast at shadow/building boundaries, constant intensity roofs. method 1 only: building boundaries approximated by straight lines	detection, delineation; polygonal delineations	none	image space
McKeown (1990) *line-corner analysis, shadow-based box verification*	nadir acquisition geometry, approximate buildings by rectangular boxes, buildings brighter than shadows and average background	detection, delineation; four-sided boxes	none	image space
Venkateswar, Chellappa (1990) *line-corner analysis by assumption-based truth maintenance system*	buildings composed of orthogonal corners, nadir acquisition geometry, buildings brighter than shadows	detection, delineation; rectilinear shapes	none	image space, 3d by shadow

Table 1-2: Recent aerial image building detection approaches, 1988-1990

as regular intensity boundaries, planar intensity surfaces, or perceptually significant groupings of edges, lines, and regions.

In the area of aerial image object detection, a wide variety of techniques have been employed. Table 1-1 summarizes some of the early work in the field [23, 35, 36, 69, 77, 80, 99]. The key assumptions of each method are presented, as well as the analysis being performed and the output representation generated, whether or not the performance of the method was quantitatively evaluated, and the coordinate frame used to represent the detection results.

Most early work focused on region-growing approaches to object detection, making strong assumptions about image photometry. These research systems typically produced detection results in the form of "blobs," boundaries with no regular geometric shape which are often rough approximations of the underlying object structure. Such systems only succeed when object boundaries have good contrast with their surroundings, an assumption that is frequently violated in urban and suburban imagery due to surface material homogeneity across surfaces and shadow occlusions on dark surfaces. Much of this early work was primarily concerned with semantic scene segmentation (detection and recognition)

prior research	acquisition, shape, and photometric assumptions	system analysis; result type	quantitative evaluation	coordinate space
Fua, Hanson (1991) *information-theoretic objective function optimization for boundary delineation*	nadir acquisition geometry, no shadow occlusions, planar intensity surfaces, no inter-reflection effects, buildings modeled by rectilinear shapes	detection, delineation; rectilinear shapes	none	image space, 3d by stereo
Bro-Nielsen (1992) *4 methods:* *1. Bayesian, CART, K-means classification* *2. shadow detection* *3. texture analysis: cooccurrence, fractal, simultaneous auto-regressive, and Fourier analysis* *4. region growing in color images, classification by region features*	nadir acquisition geometry, Lambertian surfaces, good contrast in textured areas and on shadow/building edges	detection; blob	evaluation on 1 classification image	image space
Shufelt, McKeown (1993) *merge boundaries from shadow-based and line-corner based analysis*	inherit restrictions from Irvin/McKeown (1989) and McKeown (1990)	detection; coarse polygonal boundary approximation	pixel-based metrics on 23 scenes	image space
Lin, Huertas, Nevatia (1994) *line-corner analysis, perceptual grouping for box generation, shadow verification*	buildings composed of rectangular structures, buildings brighter than shadows	detection, delineation; four-sided boxes	none	image space, 3d by shadow
McGlone, Shufelt (1994) *line-corner analysis with vanishing-point geometric cues*	buildings approximated by rectangular volumes, buildings brighter than shadows and average background	detection, delineation; flat and peaked rectangular volumes	pixel- and voxel-based metrics on 20 scenes	image space, object space 3d by vertical lines
Jaynes, Stolle, Collins (1994) *line-corner based perceptual grouping, graph search for polygons*	buildings composed of orthogonal corners, all buildings lie on same orthogonal grid	detection, delineation; four-sided boxes	evaluation on 2 images by vertex and rooftop percentage	image space
Collins et al. (1996) *line-corner based perceptual grouping, graph search for polygons*	inherit restrictions from Jaynes/Stolle/Collins (1994)	detection, delineation; four-sided boxes	evaluation on 4 images by 2d and 3d polygon matching	image space, object space by multi-image matching
Lin, Nevatia (1996) *line-corner analysis, perceptual grouping for box generation, shadow and wall verification*	building composed of rectangular structures, buildings brighter than shadows, locally weak perspective projection	detection, delineation; four-sided boxes	pixel-based metrics and manual comparison on 6 scenes	image space, 3d by shadow and wall mensuration

Table 1-3: Recent aerial image building detection approaches, 1991-present

rather than modeling scene structure (delineation). Little quantitative evaluation of results was done, and most of the early work summarized here did not attempt to model 3D structure.

Tables 1-2 and 1-3 summarize some of the recent work in the field, from 1988 to the present [13, 22, 30, 42, 46, 47, 56, 55, 57, 67, 68, 74, 93, 101]. Most of these methods are based on line and corner intensity analysis, approximating object structure by polygonal shapes (often compositions of rectilinear shapes, or even disjoint rectangles). Some methods attempt to locate shadows by thresholding for dark regions or by detecting light-dark transitions on hypothesized building edges.

prior research	image type and #	shape complexity	super-structure	building density	shadow occlusion	building occlusion	roof type
Nagao, Matsuyama (1980)	5 RGB/IR	box	no	many	no	no	flat, peak
Tavakoli, Rosenfeld (1982)	5	composite	no	many	no	no	flat
Conners, Trivedi, Harlow (1984)	1	composite	yes	many	no	no	flat
McKeown, Denlinger (1984)	6	composite	yes	few	no	no	flat
Herman, Kanade (1986)	2	complex	yes	many	no	no	flat
Harwood, Chang, Davis (1987)	5	composite	no	many	no	no	flat
Nicolin, Gabler (1987)	2	composite	no	many	no	no	flat
Huertas, Nevatia (1988)	2	box	yes	few	no	no	flat
Irvin, McKeown (1989)	2	composite	yes	many	yes	no	flat
Mohan, Nevatia (1989)	6	composite	yes	few	no	no	flat
Liow, Pavlidis (1990)	5	complex	no	many	no	no	flat
McKeown (1990)	2	composite	yes	many	yes	no	flat
Venkateswar, Chellappa (1990)	1	composite	yes	many	no	no	flat
Fua, Hanson (1991)	11	composite	yes	few	no	no	flat
Bro-Nielsen (1992)	8 + 3 RGB	complex	yes	many	yes	no	flat, peak
Shufelt, McKeown (1993)	50	complex	yes	many	yes	no	flat, peak
Lin, Huertas, Nevatia (1994)	9	composite	yes	few/many	no	yes	flat
McGlone, Shufelt (1994)	20	composite	yes	many	no	yes	flat, peak
Jaynes, Stolle, Collins (1994)	3	composite	yes	few	no	yes	flat
Collins et al. (1996)	4	composite	yes	few	no	yes	flat
Lin, Nevatia (1996)	13	composite	yes	few/many	no	yes	flat

Table 1-4: Categorization of aerial test images for building detection

Almost all of these recent methods make two key assumptions to constrain the interpretation of low-level edge and line information: the image has a nadir acquisition geometry, and building structure in the scene is composed of rectangular shape. Given these assumptions, detection of right-angled corners in the image becomes a key processing phase, as these corners are the primitive features from which structural hypotheses are constructed. If the acquisition geometry is oblique, then right-angled corners in object space no longer correspond to right-angled corners in image space; if building structure is not rectilinear in nature, then right-angled corners have little utility as primitives for object extraction. It follows that these methods will encounter difficulties on images with an oblique acquisition geometry or non-rectilinear object structure.

The output contours or volumes of these systems are typically polygonal approximations of object structure in the scene, a qualitative improvement over region-growing boundaries, although the majority of this work was not quantitatively evaluated. Most of this work produced 2D polygonal boundaries in image space, although some systems produced 3D building models in image space coordinates by shadow estimation or stereo processing. Only recently have methods been developed to produce 3D building models in object space [21, 67].

In general, the research on aerial image object detection has suffered not only from a lack of rigorous quantitative evaluation, as illustrated in the preceding tables, but also from the failure to test systems on a large representative body of images. Table 1-4 categorizes the number, type, and complexity of the test images presented in the literature. The shape complexity is described as "box," "composite," or "complex," according to whether the most complex shape in the images was a four-sided box, a composition of rectilinear boxes, or a non-rectilinear shape, respectively. Superstructure refers to the presence of texture, intensity patches, holes, or small objects atop the building that can cause interpretation difficulties. Building density is a qualitative assessment of the number of buildings in each test scene; more buildings implies more possibilities for misinterpretation, due to adjacency of buildings and similar alignments. In cases where a system was run on small pieces of an image containing few buildings to avoid search difficulties, the building density is considered to be "few." Shadow occlusion refers to shadows which drape over building objects; building occlusion refers to situations where buildings partially hide other buildings from view, including self-occlusions for composite or complex buildings. There were two dominant types of roof shape in the test images, flat and peaked.

The table shows that performance of most research systems has only been qualitatively demonstrated on a small number of test images, with no shadow or building occlusion effects. The lack of occlusion effects is partially the result of using nadir test imagery; oblique images have not seen wide use. Most systems have only been tested on images with objects which are well-modeled by composite rectilinear structures, and some have only been run on images with low building density to avoid the combinatorics of line-corner graph search for building hypothesis formation.

There exists another class of systems not represented in these summaries. These systems are semi-automated, using image features and object geometry to guide manual modeling of 3D objects [11, 29, 33, 37, 54]. While such systems can speed up the modeling process, they have known drawbacks. In particular, modeling complex building shapes is still tedious even in semi-automated systems, and boundary optimization methods often encounter local minima problems in complex imagery. Hsieh has performed a thorough analysis and discussion of design and evaluation issues in semi-automated feature extraction [40].

1.2. An approach for generic object detection and delineation

Environmental regularity plays a significant role in the perception of objects. Recent theories [5, 7, 58, 83] suggest that the apparent complexity of the environment is the result of the composition of a few basic volumetric forms, or primitives, into a multitude of different shapes. Under these theories, object detection and delineation can be reduced to two problems: detection of the primitives, and inference of the connectivity of related primitives. Such a scheme is advantageous for the detection of 3D structure, in addition to the semantic interpretation benefits it provides for recognition [96]. Rather than requiring a system to infer complex structure directly from image data, it only requires the ability to detect instantiations of a few basic primitive shapes, while still providing the system with representational power by combination of primitives.

Detection of such primitives is still an open research problem, however. A variety of volumetric primitives have been used for object modeling, among them generalized cylinders [1, 8, 14, 79], superquadrics [83, 95], and active or physically-based models [84, 100]. While these primitives have substantial descriptive power, there are difficulties in extracting them reliably from image data. Some systems require the use of range data to avoid difficulties in acquiring passive depth data; others require strong initial conditions or user guidance to ensure that optimization converges on the correct solution [24]. Even simpler geon-based primitives [5, 7] can be difficult to recover from real imagery without global information to overcome shadow and occlusion effects.

The modeling of perspective geometry can provide the necessary global information to guide the detection and delineation of volumetric primitives. Vanishing points and lines provide powerful hints for the presence of 3D structure [4]. In the vision community, the utility of photogrammetry for exploiting perspective geometry has been largely unexplored [71]. Using photogrammetry to derive the image acquisition parameters, the generation of plausible 3D object-space rectangular and pentagonal prisms for building hypotheses in monocular aerial images has been demonstrated [67].

This earlier work did not attempt to address the verification problem of selecting hypotheses which are globally consistent with the image, leaving an open question: how can interpretations be chosen which are consistent not only with local image properties, but global image properties as well? Geometric and photometric cues can provide methods for attacking the verification problem. Image cues such as junction geometry, intensity profile coherence, and shading gradients are powerful aids for structural interpretation [108]. Corner and shadow detection have proven useful in aerial image analysis, despite the lack of 3D modeling of these image features in work to date.

The techniques described and evaluated in later chapters show that rigorous modeling of scene geometry, in addition to providing a useful foundation for feature extraction, allows the use of geometric and photometric analysis for the detection and delineation of objects in aerial images. The combination of perspective geometry and structural cues provides a powerful approach for extracting 3D primitives from complex images. In the next section, we consider potential approaches for exploiting these cues.

1.3. The role of geometry and structural cues

The integration of perspective geometry and structural cues impacts processing in a broad fashion. There exist a variety of ways in which this knowledge can impact traditional image analysis techniques. The potential approaches for augmenting 3D object detection and delineation capabilities described in the following items all share a common theme: the use of perspective geometry, structural cues, or both, to provide additional information for the detection, verification, and refinement of 3D objects. While only a subset of these items will be considered in this work, all of these will prove useful in a complete 3D object detection and delineation system.

- Projective geometry permits the labeling of edges and lines as horizontals in object space, by computing azimuths of edges in object space under the assumption that they lie on a horizontal plane. A similar approach is possible for edges oriented along other planes in object space, by histogramming azimuths of lines assumed to lie on non-horizontal planes. These cues should provide additional leverage for detecting non-rectangular polyhedral primitives.

- The combination of perceptual grouping methods [58, 74] with perspective geometry will provide a powerful approach for handling typically fragmented low-level image data produced by edge detection methods. If a set of low-level edge features lie on a vanishing line, the corresponding connecting segment can be hypothesized. Further, noisy edge data can be forced to align with vanishing lines, if nearby contour information mutually supports this alignment; traditionally, noisy edge data is discarded, under the hope that sufficient edge structure can be found elsewhere to infer boundary shape. This approach allows a system to make use of noisy data, a common problem for complex images.

- Given the light source position (for aerial images, this is easily derivable from geographic coordinates and imaging date/time information), it becomes possible to analyze shadows in object space. Shadow information can be used for detection of 3D structure, by extrapolating from shadow boundaries; it can also be used for the verification of 3D structure that has been derived by other analyses. A theoretical solution for surface orientation of polyhedra and generalized cylinders based on shadow geometry has been developed [90], but it assumes that shadows have been identified and that shadow correspondence with objects is known.

- Shadow edges can be detected either by statistical estimation of a threshold [46] or by intensity correlation across edge boundaries, a qualitative photometric cue [107]. Given solar azimuth, a horizontal vanishing point can be computed for shadow lines cast by vertical object boundaries. Once shadow boundaries have been located, we can use a 3D object space adaptation of the techniques described in [46, 67] to hypothesize structure; in addition, vertical lines can be verified by searching for the corresponding shadow line.

- Given 3D object hypotheses, the expected shadow boundaries can be computed by projection from the light source, through boundary points, to the shadowed surface. These predicted shadow boundaries can then be compared with edge and intensity data to verify their existence. Such a method can, in principle, also handle shadows falling on other objects, if all object hypotheses are considered simultaneously. Further, this shadow boundary prediction process has a role in refinement; shadow mensuration provides another estimate of structure height.

- Knowledge of the light source position also permits photometric analysis. Primitives with curved surfaces, such as spheres and cylinders, can be coarsely evaluated to verify the expected shading effects. For adjacent planar surfaces, light-dark transitions can be verified to check photometric consistency, and trihedral vertices in object space can be analyzed for the appropriate intensity transitions in image space. Although surface material changes across surfaces cannot be inferred from panchromatic imagery, coarse photometric cues are still useful for the perception of 3D structure [108].

- The scene geometry can be used for visibility analysis. Given a set of 3D object hypotheses for a scene, which may have object occlusions from the camera viewpoint, visibility algorithms can predict which surfaces should be visible (and hence verifiable) by image analysis. There has been much work in the computational geometry, visual part inspection, and motion planning literature on visibility analysis; references for this work can be found in [85].

An interesting aspect of the research ideas presented here is their independence of process flow. The integration of cues derived from perspective geometric and photometric cues can be applied to a variety of intermediate structural representations, regardless of how those structural representations are being used to derive 3D primitives from an image.

Another important aspect of these ideas centers on recognition. Although the primary focus of these approaches is 3D primitive detection and delineation, the use of object space for geometric and photometric reasoning allows the derivation of intrinsic properties for objects, such as height, width, and length in world coordinates, the slopes of surfaces, and the presence of superstructure as modeled by the chosen primitives. These intrinsic properties are well-suited for use by existing image interpretation techniques [72].

The research ideas described here are essentially monocular in nature, but they do not preclude the use of stereo. Stereo vision can play at least three roles in 3D scene analysis: to determine background depth and to refine height for hypothesized objects; to provide verification and refinement information for features by matching edges, lines, and corners across images [89]; and finally, to form the basis for a closed-loop analysis system which can use feedback to converge on the correct 3D model for a scene.

The latter conjecture is the most interesting (and the most speculative). With a model of imaging geometry for two or more images of a scene, and an initial 3D model of the scene, it is possible to use texture mapping to transform one image, *A*, from its viewpoint to the viewpoint of another image of the same scene, *B*. This texture-mapped image and image *B* could then be input to a stereo process; the resulting disparity map should represent a flat plane, since the images will be identical if the 3D model was correct. Any deviations from the plane represent modeling errors, and serve as feedback to correct the 3D model, creating a closed-loop system. While the incorporation of stereo data and other depth-map information for 3D object extraction is beyond the scope of this thesis, it may ultimately be needed in a supervisory role to merge information derived from multiple views.

1.4. Main contributions of the thesis

The research ideas outlined in the previous section illustrate the variety of ways in which rigorous modeling of imaging geometry permits object detection and delineation to take place, even in unconstrained environments or in the absence of explicit object models. This thesis focuses on a subset of these ideas and applies them to the domain of object detection and delineation in aerial images, leading to the following contributions:

- Utilization of simple primitive volumetric models as the representation for objects. This approach gives representational power by combination of primitives, avoiding the need for a large model library and providing flexible modeling of complex shapes.
- Approaches for exploiting perspective geometric cues for vanishing-point analysis and junction geometry analysis. Traditionally, perspective information has been used to derive camera pose; by reversing the problem, perspective information can be used for feature extraction, and in particular, primitive extraction.

- Approaches for using projective geometry to analyze shadow shape and photometry. With a rigorous photogrammetric foundation, shadow shape of 3D structures can be predicted for verification, or be derived from low-level image data to infer the presence of objects; photometric cues across surface orientation discontinuities can also be employed for verification.
- Quantitative evaluation of the generic object extraction methods on a wide variety of complex aerial images. As noted in Section 1.1, quantitative evaluation in aerial image analysis has been infrequent at best, and even qualitative analysis has been limited by the use of small data sets. This work is evaluated rigorously against manually-derived 3D models of several test scenes, providing a firm basis for future work.
- An illustration of the benefits of photogrammetric modeling to some computer vision tasks. Traditionally, the computer vision community has neglected the use of precise image acquisition information [71]. Its use should be viewed as broadly applicable in other vision domains where precise modeling of viewing geometry can facilitate object analysis [53], and it should also be viewed as a rigorous mechanism for analysis in world-centered coordinates, a necessity for automated cartography.

The last item is perhaps the most significant. Although this thesis is concentrated on object detection and delineation in aerial image analysis, there is no intrinsic restriction in photogrammetric methods that prohibits their application to other computer vision domains. Perspective geometry is frequently viewed as a mathematical difficulty; it should instead be regarded as a valuable source of information for scene analysis.

Chapter 2
Object Detection and Delineation

As discussed in Chapter 1, much of the work on object detection and delineation relies heavily on a variety of assumptions about the scene, the objects in it, or the imaging process. Systems operating under these constraints are able to achieve reasonable performance on the limited class of imagery which obeys these assumptions. Unfortunately, such systems typically fail on imagery outside of the specific domain for which they were designed, and it is often unclear how the assumptions for these systems can be removed in a principled way to achieve improved performance on a wider set of imagery, without sacrificing performance in the original domain.

The ultimate goal of object detection and delineation research is to produce an automated system which can achieve qualitatively and quantitatively robust performance on images with a wide variety of viewing angles, object shape complexity, object density, object occlusions, and shadow effects. All of these factors must be explicitly taken into account in any system that achieves this goal. While such a system lies beyond the current state-of-the-art, analysis of the successes and failures of the research to date leads directly to the principles that will guide its development.

In this chapter, a set of principles for object detection and delineation is presented. Arguments for these principles, based on prior research and some examples from complex aerial imagery, are also given. Following a discussion of these principles, a brief outline of the system described in this work is presented, setting the stage for later chapters.

2.1. Modeling image geometry

Perhaps the simplest possible representation of classical computer vision problems involves three abstract entities: the environment under consideration by a *sensor*, known as the *scene*; the data produced by that sensor through an *image acquisition* process, known as the *image*; and the internal representation of the scene produced by analysis of the image, known as the *scene model*. The goal of a computer vision system is to generate a scene model from an image, such that the scene model accurately and precisely represents the scene, according to predefined performance criteria. Figure 2-1 illustrates the problem.

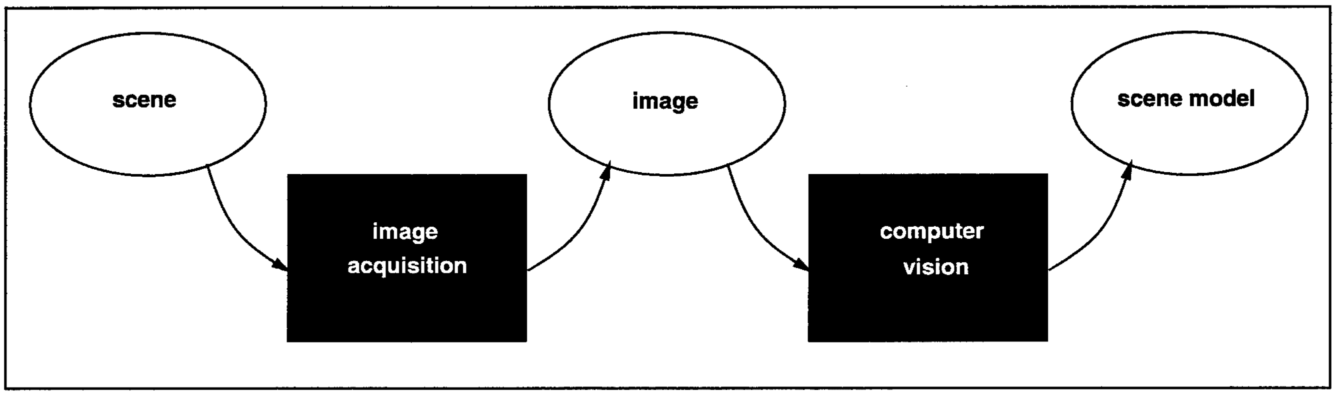


Figure 2-1: The computer vision problem

Before proceeding further, it is useful to instantiate these terms for the problem domain to be explored in this thesis. For the purposes of this work, a *scene* will be some geographic location, imaged by a frame mapping camera from an aerial platform. An *image* will be the digitized result of the imaging process, and the *scene model* will be a 3D wireframe representation of the manmade structures in the scene, referenced to geographic coordinates. Chapter 6 describes the evaluation of the accuracy and precision of the scene model in detail; for the discussion here, it suffices to state that the goal is to match as closely as possible a manually constructed scene model for the geographic location in question. Finally, the *sensor* to be used exclusively in this work is a traditional frame mapping camera; more will be said about alternate sensors in Section 2.6.

It is well understood that the 2D image is an underconstrained representation of the 3D scene. In order for a computer vision system to generate a scene model from the image, it must utilize some form of external knowledge to constrain the search space for scene models. This knowledge takes the form of constraints which often restrict some aspect of the scene, the image, or the scene model. Examples of constraints for each of these entities include:

- **scene constraints**: assume low object density, specific object types, and/or specific illumination conditions
- **image constraints**: assume specific viewing angles and/or image resolutions
- **scene model constraints**: assume scene can be modeled by 2D shape descriptions, by image space descriptions instead of object space descriptions, and/or by explicit models

The advantages of these constraints are clear: they give a system leverage for attacking a mathematically ill-posed problem, and greatly limit the size of the hypothesis space which must be explored to produce a scene model. The disadvantages are equally clear: any of these constraints limits the generality of a vision system, and since the use of these constraints is often integral to a system's performance, generalizing or removing them is frequently an intractable task.

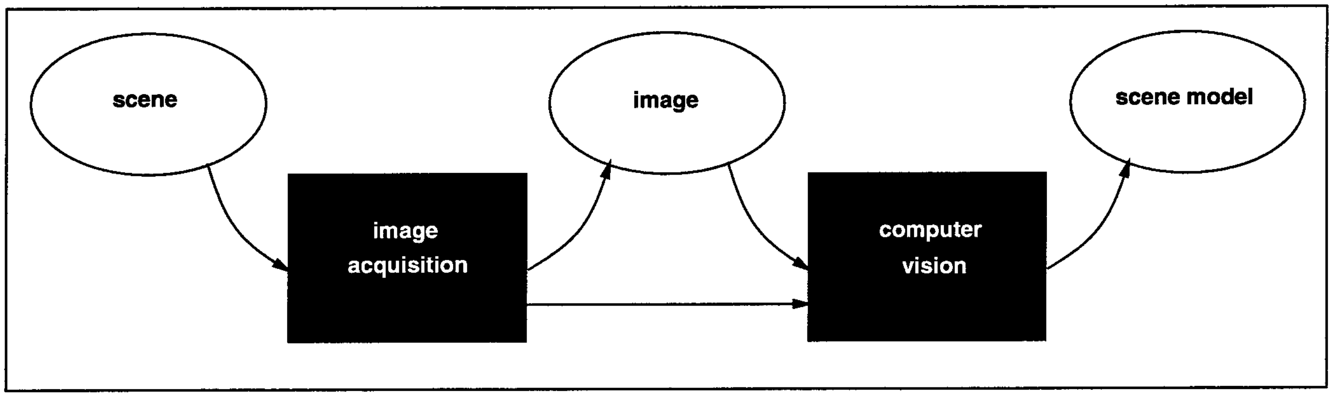


Figure 2-2: Using image acquisition information

This leads to an unfortunate tradeoff: as the generality of the vision domain increases (and the number of constraints on the scene, image, or scene model decreases), the hypothesis space for scene models increases as well. Avoiding this tradeoff constitutes the key problem for generic object detection and delineation. To solve the problem, we must find constraints which obviate the need for an exhaustive search of hypothesis space, yet place minimal restrictions on the scene, image, and scene model.

A natural source for constraints that satisfy these requirements is the image acquisition process itself (Figure 2-2). Image acquisition imposes a geometry on the mapping of scene measurements to image pixels. Modeling this geometry gives a vision system the ability to constrain its search space, by ensuring that all interpretations of image data are legal in the geometry. The virtue of constraints derived in this fashion is that they place no restrictions on the scene, image, or scene model; the geometry is intrinsic to the imaging process, and any accurate scene model, independent of its representation, must obey the same geometric constraints. This idea forms the first of several principles for generic object detection and delineation:

PRINCIPLE 1: **Rigorously model the image acquisition process.**

Following this principle provides several benefits for an object detection system: it allows the computation of vanishing points, which suggest plausible orientations for the projections of lines in the scene; it permits analysis from arbitrary viewing angles; and, it gives the ability to manipulate data directly in 3D object space, using scene-centered coordinates.

Recently popular methods which avoid full modeling of the image acquisition process involve the use of geometric invariants, properties of image features which remain constant over changes in viewpoint. Geometric invariants have been employed to address problems in camera calibration, model-based object detection, and space reconstruction. Thorough surveys of this work can be found in [75, 76, 103].

However, many of these methods only recover structure up to a 3D affine transformation from an uncalibrated camera or set of cameras, and require known point correspondences. An invariant-based method has been developed to reconstruct Euclidean space from uncalibrated cameras [34], but this method also requires known point correspondences, and may be unstable under certain camera rotations.

Since photogrammetric methods also require point correspondences for resections, it is natural to consider the performance characteristics of invariant-based methods with their photogrammetric counterparts.

In recent studies [2, 3], the performance of invariance techniques and photogrammetric methods were compared on problems of 2D and 3D point transfer, and object space point determination for 3D objects. For 2D point transfer, results were comparable. However, for 3D point transfer, invariant methods were numerically unstable; photogrammetric least-squares solutions with constraints were robust, although the differences decreased with increases in the number of points. For object space point determination, results showed that photogrammetric position estimates were superior by two orders of magnitude and less sensitive to object point configurations. These studies show that precise object delineation is best achieved by the use of rigorous photogrammetric methods. Invariant approximations are susceptible to numerical stability problems, produce substantial positional error in object space point determination, and can be sensitive to certain camera rotations.

Of course, invariants provide a useful tool in situations where the interior orientation of the camera is not known; for uncalibrated cameras, it is possible to recover 3D space up to an affine transformation. Depending on the problem domain, this level of modeling accuracy may well be sufficient. For simple robotic tasks in which a manipulator is guided through its environment by a visual sensor, an invariant-based approximation to the true camera geometry might well be sufficient for avoiding obstacles and manipulating target objects. Likewise, the solution of robotic navigation problems may not need, or even benefit, from precise 3D reconstructions of the environment; a qualitative sense of 3D structure could be sufficient to enable a robot to move through its surroundings without difficulty.

Nevertheless, for certain domains, including the cartographic feature extraction application explored in this thesis, proper camera modeling avoids the problems associated with invariant solutions, and gives the ability to work in object space directly. This is crucial for reasoning about effects which are intrinsically three-dimensional, such as shadow casting and object occlusion.

2.2. Primitives: generic object models

Assuming that Principle 1 is followed, a vision system will have at its disposal a source of geometric cues which do not place restrictions on the scene, image, or scene model. Further, it will have the ability to perform measurements and analysis in object space, which will be useful for attaching scene models to world coordinates, as well as reasoning about 3D effects. However, Principle 1 does not dictate the choice of scene model representation; it simply provides the tools for constructing instances of that representation. The second principle, briefly mentioned in Section 1.2, makes this representation choice:

PRINCIPLE 2: Use primitives for generic shape representation.

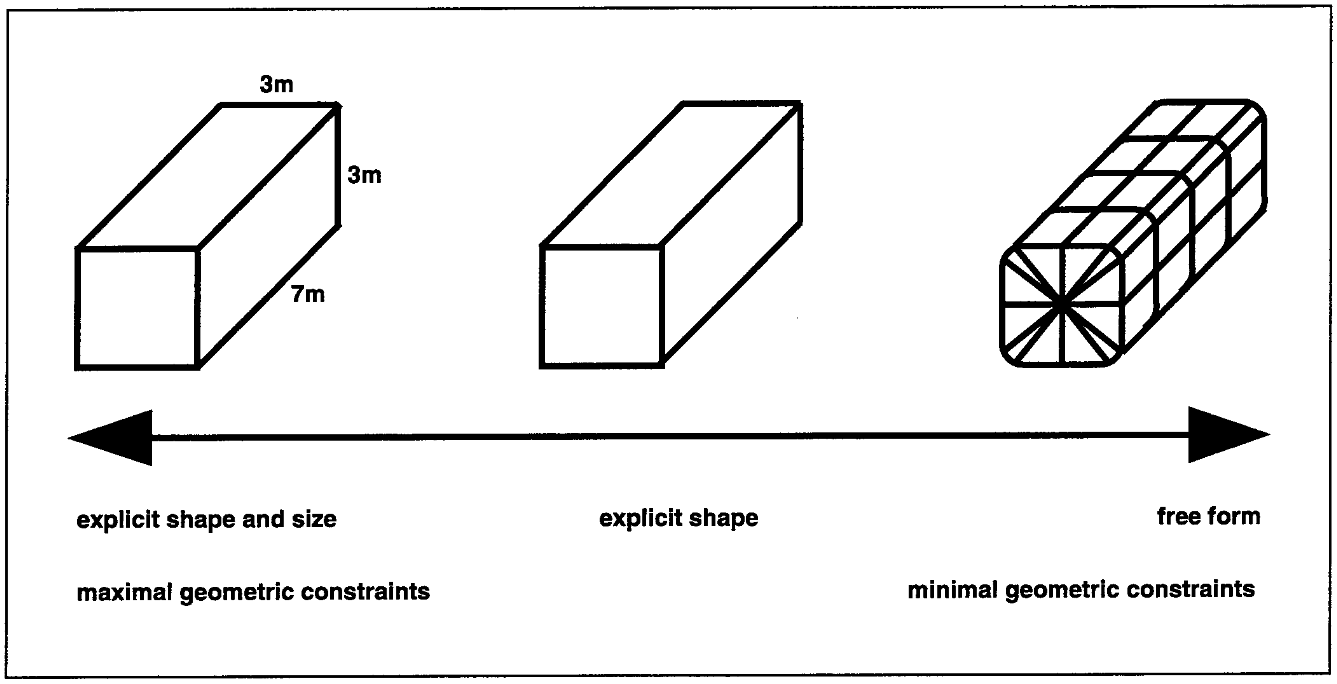


Figure 2-3: The 3D object representation spectrum

It is well understood that the choice of object representation in vision problems is crucial to system performance. A great deal of work has been done in exploring the advantages and disadvantages of various representation schemes, along a continuum of choices which vary primarily in the extent to which they explicitly represent shape and size. Figure 2-3 depicts this continuum, illustrating a general spectrum of choices which can be distinguished by the degree to which they are geometrically constrained. CAD models and deformable models lie at the left and right extremes of the spectrum, respectively; primitives lie in the middle.

Much of the work on object detection and delineation has used explicit models, where the shape and size of the objects is known. Model-based methods use these explicit models as both the scene model representation and a source of geometric cues to constrain the search for valid instantiations of these models. When objects of interest can be accurately modeled, these methods can be very effective in hypothesizing and verifying scene models [6, 9, 16, 31].

In the general situation, however, full knowledge about object structure may not be available. One option is to compile that knowledge when necessary, but this can be intractable for domains where the size of the model library can potentially be very large. One such example is the domain of aerial image analysis, where manmade structures occur in a wide variety of shapes and sizes. Manual compilation of a comprehensive model library for this domain would be extremely time-intensive, at best.

Although it may seem paradoxical to attempt to detect and delineate objects when those objects are not explicitly known, humans already display this ability routinely. Common objects in our environment, such as chairs and tables, can be readily detected and delineated by humans even though their specific sizes, shapes, surface patterns, and material composition all exhibit extensive variations. Further, even when presented with objects which have no known semantics, humans are nonetheless able to describe these objects in terms of basic parts, such as blocks, cylinders, and cones [7].

It has recently been argued [5, 7, 58, 83] that the complexity of our visual surroundings is the result of the composition of a few basic volumetric forms, known as *primitives*. The representational power of primitives for vision is analogous to that of phonemes in speech. In both cases, a small set of basic elements are used to construct arbitrarily complex elements. Another virtue of this representation is that it is naturally suited for recognition; the connections between primitives can be used to infer semantic interpretations [96]. It has also been argued that such representations can even represent natural forms, by perturbing primitive surfaces with fractal models [83].

The drawback of this approach lies in the absence of robust, general techniques for extracting primitives from imagery. A variety of volumetric primitives have been used for generic object modeling, including generalized cylinders [1, 8, 14, 79], superquadrics [83, 95], and active or physically-based models [84, 100]. All of these representations have substantial descriptive power, but there have been difficulties in reliably extracting them from images. Some systems require range data to avoid potential stereo correspondence and matching problems; others require strict initial conditions or user guidance for robust optimization [24]; others encounter problems in handling shadow and occlusion effects.

The problems with these approaches do not lie in the choice of representation (although certain choices may be more natural than others, depending on the domain); they lie in the absence of powerful constraints for detecting and delineating primitives which do not restrict system applicability. Principle 1 provides the solution: use constraints and cues derived from rigorous photogrammetric modeling to guide the search for primitives.

2.3. Bounding hypothesis space

The principles stated thus far derive from general considerations of the vision problem. Principle 1 leads to constraints which avoid restrictions on the scene or the image, and Principle 2 leads to a representation which avoids restrictions on the scene model. The combination of these principles gives a coarse picture of the key elements of a generic object detection and delineation system. However, there are other principles the system should obey, which derive from specific considerations of the details of the problem. In particular, observations and examples from earlier research approaches highlight practical aspects of object detection which must be addressed in a successful implementation of a general system.

The next pair of principles are related. Together, they set the boundaries within which a generic object detection system can effectively and accurately search for hypotheses. The first of these principles sets an upper bound on the size of the hypothesis space:

PRINCIPLE 3: **Apply geometric constraints as early and often as possible.**

The idea that geometric constraints should be employed at every opportunity to reduce the search space is not new. A great deal of work has been performed on representations of hypothesis space known as *interpretation trees* [32]; the key insight of this work is that geometric constraints can efficiently and accurately prune large sections of these trees which need never be visited. By doing this pruning incrementally, as each new data feature is considered as a possible addition to a set of active hypotheses, erroneous hypotheses can be removed as soon as they violate geometric constraints. This approach has also been used with a variety of geometric constraints and sensor types [45]. A thorough discussion of this research area can be found elsewhere [31].

A common theme of this research is reliance on explicit object models to derive geometric constraints. If Principle 2 is followed, this necessarily implies that reliance on explicit information must be weakened, since the specific geometries of object primitives will vary from instance to instance. However, certain geometric constraints can still be employed, depending on the primitive in question. For instance, a generic rectangular volume primitive has no geometric constraints on size, since its length, width, and height are variable; but it does have constraints on angles at corners (all angles are right angles) and on line orientations (opposite sides of faces are parallel).

It is worth noting that model-based geometric constraints are quantitative in nature; they express explicit relationships between model features, such as the expected length of a model edge, or the expected relative angle formed by two model edges. These are not the only possible geometric constraints available; other constraints are qualitative in nature. The seminal examples of such constraints come from line-drawing understanding research. Classical line labeling and junction analysis techniques [17, 43, 48, 62, 102] express constraints in terms of expected vertex/edge concavity/convexity, line or surface visibility, or global geometric relationships such as symmetry, parallelism, or perpendicularity.

The constraints we will derive from rigorous camera modeling in later chapters combine aspects of both types of constraints. They will express allowable line orientations in terms of quantitative measurements known as *vanishing points*, but they will also share aspects of qualitative line labeling approaches to incrementally construct geometrically valid hypotheses for primitive volumes. In either case, they share the same goal: efficient and accurate pruning of hypothesis space.

Principle 3 was described as setting an upper bound on the size of the hypothesis space for scene models, by limiting the number of false hypotheses via geometric constraints. Its counterpart, Principle 4, sets the lower bound:

PRINCIPLE 4: Do not discard interpretations prematurely.

On the surface, this may seem like an obvious principle for any object detection and delineation system. It is clear that any system which intends to achieve robust generation of scene models must not eliminate potentially correct interpretations of image data, particularly if those interpretations happen to be the only correct ones. Nonetheless, the application of this principle to the general vision process leads to some interesting and less obvious conclusions. We first turn to an active topic of research, perceptual grouping.

Perceptual grouping methods are based on Gestalt theories of human vision, which suggest that preferred interpretations of visual data are those which explain it most simply. This notion of simplicity was qualitatively defined in terms of a variety of relationships among image features, such as proximity, similarity, continuation, closure, symmetry, and familiarity. The modern interpretation of these theories, best elucidated in [108, 58], hinges on a probabilistic argument; the significance of a potential grouping is inversely related to the probability that it could have arisen by accident.

For example, consider two long edge segments, lying on the same line, separated by a small gap. The *a priori* likelihood of two long edge segments sharing the same line in close proximity is small. Of course, the likelihood of two line segments sharing any specific relationship is also small. In this case, however, collinearity is viewpoint invariant, unlike many other relationships, and so this grouping can be regarded as significant. Lowe gives a detailed exposition on the determination of significant relationships [58].

The major benefit of perceptual organization techniques is that they give a system the ability to derive groupings and structures without prior knowledge of the scene, by performing local examinations of image features for significant relationships. By grouping related features, a system can reduce the hypothesis space and simultaneously constrain features to specific three-dimensional interpretations. For instance, collinear edges in 2D most probably correspond to collinear edges in 3D, barring a viewpoint singularity.

An important aspect of this approach is its dependence on local rather than global techniques. From a computational standpoint, it would be intractable to consider relationships between all possible subsets of image features. In practice, it is also the case that causally related features frequently appear in close proximity, and there is psychophysical evidence that human vision employs the same type of heuristic to limit processing complexity [58].

The difficulty with this approach, and the reason that it comes into conflict with Principle 4, is that it is often applied early in the processing stream, well before other information could be employed to verify the correctness of potential groupings. As a result, erroneous groupings can be propagated through the system at the expense of discarding the ungrouped image features, making it difficult, at best, for a system to recover from such errors. To state the issue another way: a local analysis algorithm cannot be expected to make globally correct decisions [49].

To illustrate this kind of failure, we consider a simple edge segment grouping task. In complex imagery, it is common to observe edge fragmentation along object boundaries, due to a variety of causes ranging from noise in the image to abrupt intensity transitions along the object's edge. A solution to this problem is to employ perceptual grouping methods to link fragmented edges which appear to share object boundaries. A common method is to check collinearity and segment proximity, both local relationships between edges, to determine whether segments should be grouped.

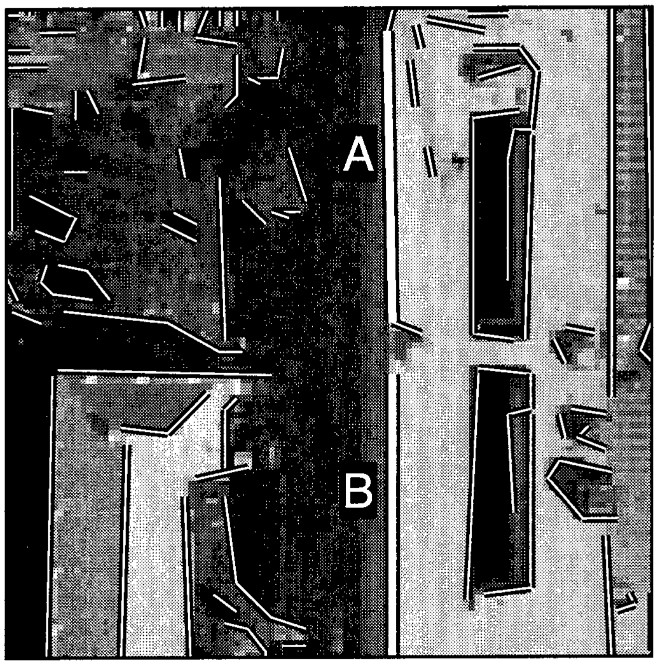


Figure 2-4: Correct edge grouping

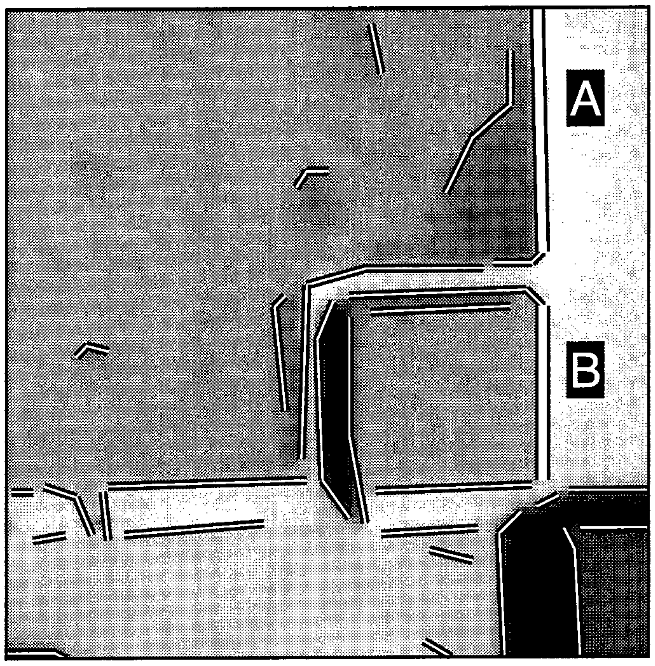


Figure 2-5: Erroneous edge grouping

Figures 2-4 and 2-5 show the results of applying the Nevatia-Babu edge operator [78] and a recursive line fitting procedure to obtain straight line segments for a pair of aerial images. Figure 2-4 shows a typical edge fragmentation effect, caused in this case by the shadow of a ventilation unit on the building rooftop. Edges A and B comprise the two fragments of the building edge. In this case, a segment grouping technique would group A and B, since they are collinear and lie in close proximity to one another. The resulting combined edge is a more useful feature for extracting the complete structure.

However, the same method can generate erroneous groupings. Figure 2-5 shows another typical situation, where two buildings lie in close proximity to one another. Edges A and B are not fragments, since they correspond to different structures in the scene. But, as in Figure 2-4, the edges are collinear and lie in close proximity, so a grouping technique would group these edges. At this point, an interpretation has been discarded. Assuming standard methods for building detection are being used, the system will now generate one hypothesis for the building corresponding to edge A, incorrectly including the building corresponding to edge B.

The key reason for the failure of this approach is its reliance on local rather than global information. Locally, there is not enough information to distinguish between the situations in Figures 2-4 and 2-5. Global analysis of line structure could prevent this mistake, but only at the expense of considering all potentially relevant edges. It is exactly this avoidance of global analysis, necessary to obviate combinatorial explosion, which also results in incorrect low-level feature generation.

This does not argue that perceptual grouping approaches should be abandoned; in fact, the detection of non-accidental relationships gives a system significant power for dealing with partial or noisy data. The key points are twofold: that grouping techniques should be applied only when there is enough information to verify the groupings they produce, and only when this information can be exploited without incurring combinatorial difficulties.

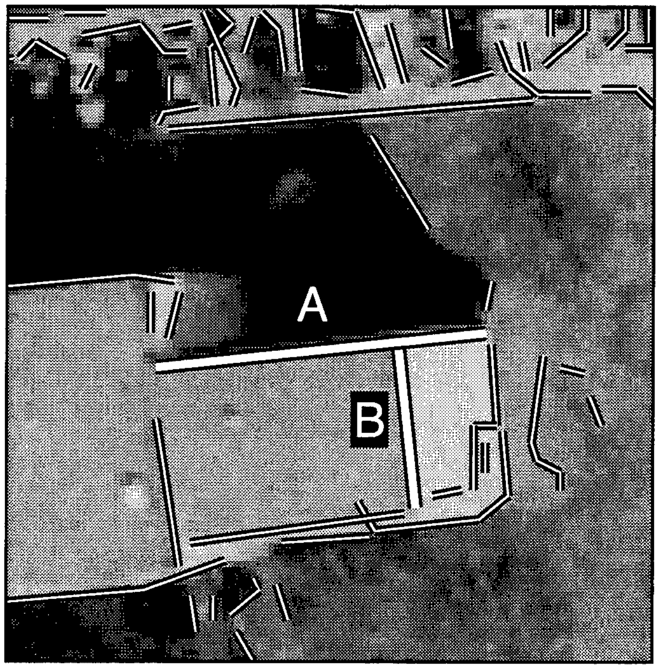


Figure 2-6: Correct T splitting

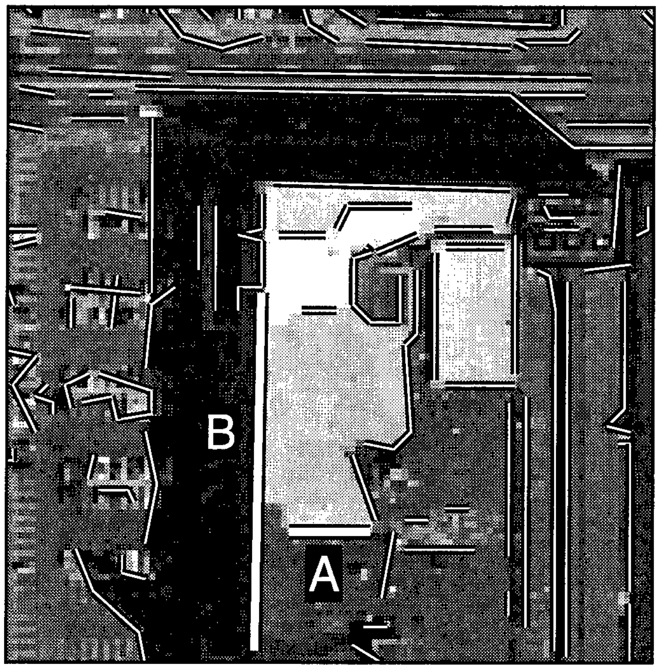


Figure 2-7: Erroneous T splitting

The use of perceptual grouping in the early processing phases of an object detection and delineation system represents one example of premature removal of feature interpretations. Principle 4 has other applications as well. A classic difficulty frequently encountered in aerial imagery involves T-intersections, formed by the intersection of one line segment with the interior of another line segment. Huertas and Nevatia [42] discussed the difficulties involved in interpreting T-intersections, and described a local heuristic for breaking the T-intersection into two corners based on the distance between the stem and arms of the T and their lengths. However, reliance on local information can fail for this interpretation task as well.

Figure 2-6 shows a building viewed at an oblique angle, in which both the rooftop and a wall of the building are visible. In this case, edge A corresponds to two edges on the structure. Edge B forms a T-intersection with edge A, and breaking at the intersection would correctly split A into two edges, one for the roof edge and one for the wall edge.

Figure 2-7 shows another building with several different roof surface materials. The change in intensity at a surface material transition on the roof of the building produces edge B, which locally appears to form a T-intersection with edge A. Edge A corresponds to a single edge of the building. In this case, a local T-intersection heuristic would split the T, creating two edge segments for one edge in the scene, discarding the correct interpretation of A. One could imagine adding intensity-based heuristics to the processing of T-intersections, but these heuristics would still be local in nature, and changes in photometry and surface material for different views and buildings would inevitably violate these heuristics.

T-intersection processing is another example of a local interpretation technique which is often applied early in the processing chain of data-driven building detection systems. Its failure in situations where global information is necessary to ensure correct analysis illustrates the importance of keeping potentially valid interpretations. Techniques which reduce the search space of possible interpretations should be applied only when sufficient information is available for verifying that the reductions have not discarded valid interpretations.

This section has described the principles defining the bounds of the hypothesis space in which a general object detection and delineation system must operate. They can be viewed as describing a fundamental tradeoff between search efficiency and interpretation accuracy. Stated more succinctly, Principles 3 and 4 combine to form a simple maxim about hypothesis space bounds: *"use geometric constraints as soon as possible, but no sooner."*

2.4. Modeling 3D effects

Another aspect of generic object detection and delineation involves the analysis of shadow and occlusion effects. These effects are dependent on the position of the illumination source, the camera viewpoint, and the placement and shape of the objects in the scene. When the position of the illumination source and the parameters of the camera model are known, then these effects can be utilized to infer the presence and shape of objects.

It is well understood that shadows are powerful cues for analysis and verification of shape; much theoretical and empirical research has been done on shadows. Shafer developed a theoretical solution for the surface orientation of polyhedra and generalized cylinders based on shadow geometry [90]. In the aerial image analysis domain, shadow analysis has been used in nadir imagery for region-based building detection [57], object/shadow corner labeling [42], and building hypothesis detection, verification, and clustering [46].

Occlusion has been utilized less often as a tool for hypothesizing scene models. While empirical shadow methods have had some success with local intensity evaluation to determine object-shadow boundaries, it is difficult to imagine how a system could correctly infer the occlusion of one object by another object using only local information about line crossings and intensity transitions. One popular approach for handling occlusions is the active contour model, or "snakes" approach [54]. By defining appropriate smoothness functionals, which are essentially global shape constraints on the contour model, these models can bridge gaps in edge data. However, these models do not make any distinction between gaps caused by intensity variations and gaps caused by actual object occlusions.

In either case, limiting approximations have been made to handle these effects. Shadow analysis for complex aerial images often assumes a nadir viewpoint, since the computation of shadow boundaries often reduces to a simple 2D projection in this case. Occlusions are typically treated as a difficulty in scene analysis, to be resolved by techniques such as active contour models; in these cases, no attempt is made to actively use occlusions to infer object structure, or even to understand the causes of the occlusions. The next principle, like Principle 1, argues for more rigorous modeling:

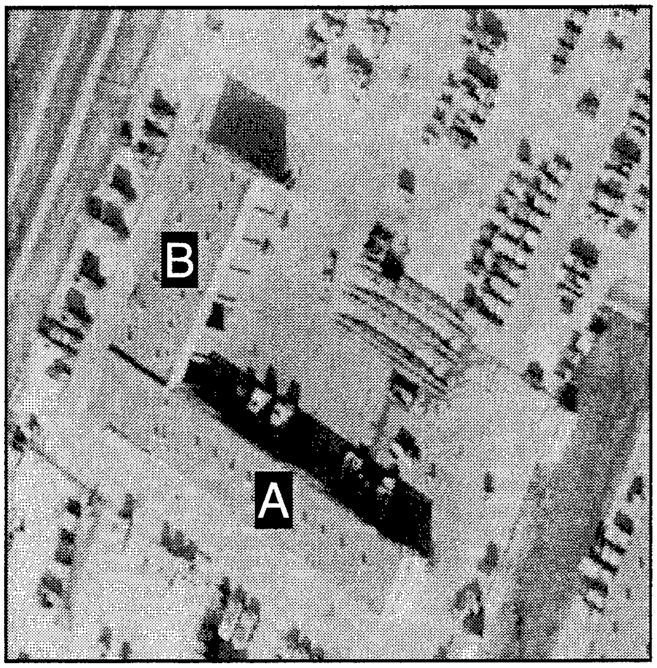


Figure 2-8: Occluded shadows

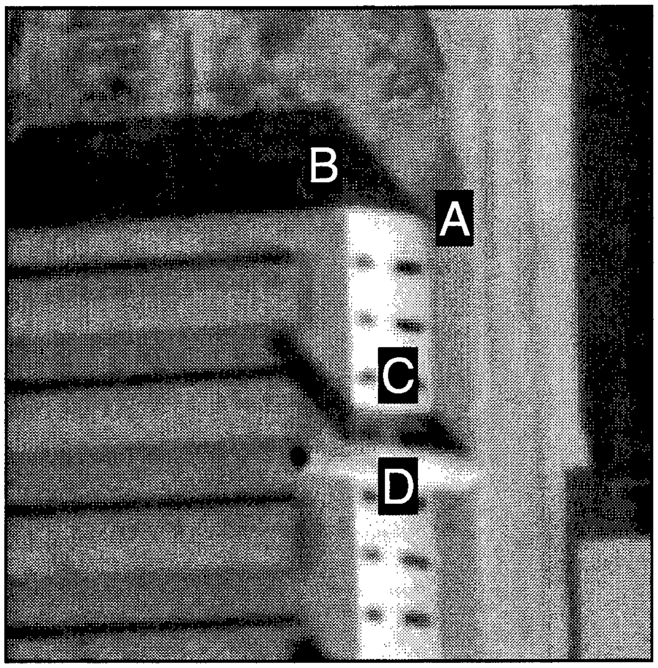


Figure 2-9: Occluded building

PRINCIPLE 5: Model 3D effects in 3D.

Shadow and occlusion effects are inherently three-dimensional, and often interact to produce complex images which violate common assumptions. Consider the L-shaped building in Figure 2-8, consisting of two wings A and B. B casts a shadow on the ground, but it partially occludes its own shadow. Wing A casts a shadow as well, but one end of the shadow falls on the wall of B, altering the shadow shape. A nadir 2D approximation of shadow projection for this image would be unable to analyze the shadow occlusion of B, or the shadow of A being draped on B, whereas full 3D modeling in conjunction with a camera model permits correct analysis.

The virtues of modeling 3D effects in a 3D space do not lie solely with the ability to avoid modeling errors. These effects also provide multiple measurements of object structure in the scene, which leads to more robust performance. Consider the situation in Figure 2-9, where a cylindrical smokestack both occludes the rectangular building and casts a shadow on it. Assuming that a system has detected both the rectangular and cylindrical volumes, it might next attempt to refine its delineations using measurements of image data.

If the system models 3D effects in 3D space, it has several measurements available for refinement. For example, the height of the rectangular building can be measured in four different ways: by measuring A, the height of its vertical edge; by measuring B, the length of the shadow of A, and using the solar elevation and B to solve for the height of the vertical; and by measuring C and D, the heights of the vertical shadows of the smokestack on the building.

While an automated analysis of this type lies well beyond the capabilities of existing techniques, it is nonetheless an illustration of the types of techniques that will be not only necessary, but advantageous, for the simultaneous and robust solution of object position. Recent work on constraint systems for aerial image resection and building hypothesis refinement offer one promising approach for carrying out these kinds of analyses [66]. In any case, even a system which completely neglects position estimation must still model shadow and occlusion in 3D space to obtain good performance over a wide range of viewpoints and illumination angles.

2.5. Evaluating performance

Unlike the principles described thus far, the next principle makes no claims about the approach that a generic object detection and delineation system should take to achieve good performance. It does argue, however, for a particular approach for achieving and evaluating that performance, and taken in that sense, it is a natural principle to include in the discussion.

Performance evaluation is often treated as a secondary aspect of the research on object detection and delineation, if it is treated at all (as Tables 1-1, 1-2, and 1-3 in Chapter 1 illustrate). This has been a significant hindrance to progress in the field, since it is difficult to judge the strengths and weaknesses of various approaches, or the relative effectiveness of new techniques on known hard problems, without serious attempts to quantify performance in meaningful terms. The next principle addresses this issue:

> PRINCIPLE 6: **Evaluate performance comprehensively and fairly.**

What is meant by "comprehensively?" In brief, it means evaluating performance on significant amounts of data. No meaningful conclusions can be drawn about an algorithm which is purported to do generic object detection and delineation, yet has been run and evaluated on only one image, or even one object. To achieve significance, it is necessary to evaluate performance on a wide variety of scenes (ensuring breadth of coverage) and on many scenes with similar content (ensuring depth of coverage). As the breadth of coverage increases, so does the likelihood that the algorithm is truly generic; as the depth of coverage increases, so does the likelihood that the algorithm is robust to small changes in scene and image content. This also has a direct and beneficial impact on the development of new algorithms. The use of a broad, deep test suite for evaluation highlights specific algorithmic deficiencies, but it can also highlight classes of imagery which present unique difficulties for detection and delineation systems.

What is meant by "fairly?" It means that the methods chosen for performance evaluation should give an accurate picture of system effectiveness. A typical problem in this regard stems from the lack of distinction between "detection" and "delineation" performance. For example, one might use a metric which considers a building to be "found" if a building hypothesis in the scene model overlaps it by at least 50%. Given that metric, a system which generated "half-buildings" for every building in a scene could be said to have perfect performance. While technically correct, such a statement is misleading in that it confounds detection and delineation. The delineation performance in this example would undoubtedly be less impressive.

2.6. System structure

The previous sections in this chapter have described a set of guiding principles for the solution of the generic object detection and delineation problem. These principles are based on the observation that a successful solution must lead to a system with the ability to operate accurately and robustly over wide variations in viewing angles, object shapes, object densities, occlusions and shadow effects. These principles are enumerated without comment in Figure 2-10 for quick reference.

1. **Rigorously model the image acquisition process.**
2. **Use primitives for generic shape representation.**
3. **Apply geometric constraints as early and often as possible.**
4. **Do not discard interpretations prematurely.**
5. **Model 3D effects in 3D.**
6. **Evaluate performance comprehensively and fairly.**

Figure 2-10: Principles for object detection and delineation

The remainder of this thesis describes and evaluates a system which was designed with the goal of adhering to these principles as closely as possible. This system, PIVOT (**P**erspective **I**nterpretation of **V**anishing points for **O**bjects in **T**hree dimensions), is a data-driven fully-automated monocular building extraction system for frame mapping photography. PIVOT takes as input a gray-scale aerial image and the image acquisition parameters (interior and exterior orientation and the date and time of acquisition), and produces 3D wireframe representations of the buildings in the image, referenced to geographic coordinates. PIVOT requires no parameter adjustment or threshold tuning, nor does it require that images be cropped so that building density is low.

Figure 2-11 presents a high-level illustration of PIVOT's process flow in the context of the general vision problem depicted earlier in 2-2. Chapter 3 addresses vanishing point analysis, Chapter 4 treats corner and primitive generation, and Chapter 5 discusses primitive combination, extension, and verification. It should be noted that this diagram is a simplification of the process flow of PIVOT; in particular, separation of hypothesis generation and verification into separate phases is not a strict one. Some verification analysis occurs during the generation process, and vice versa. For illustrative purposes, however, the diagram reflects the essential aspects of PIVOT, including its heavy reliance on image acquisition knowledge in many processing phases.

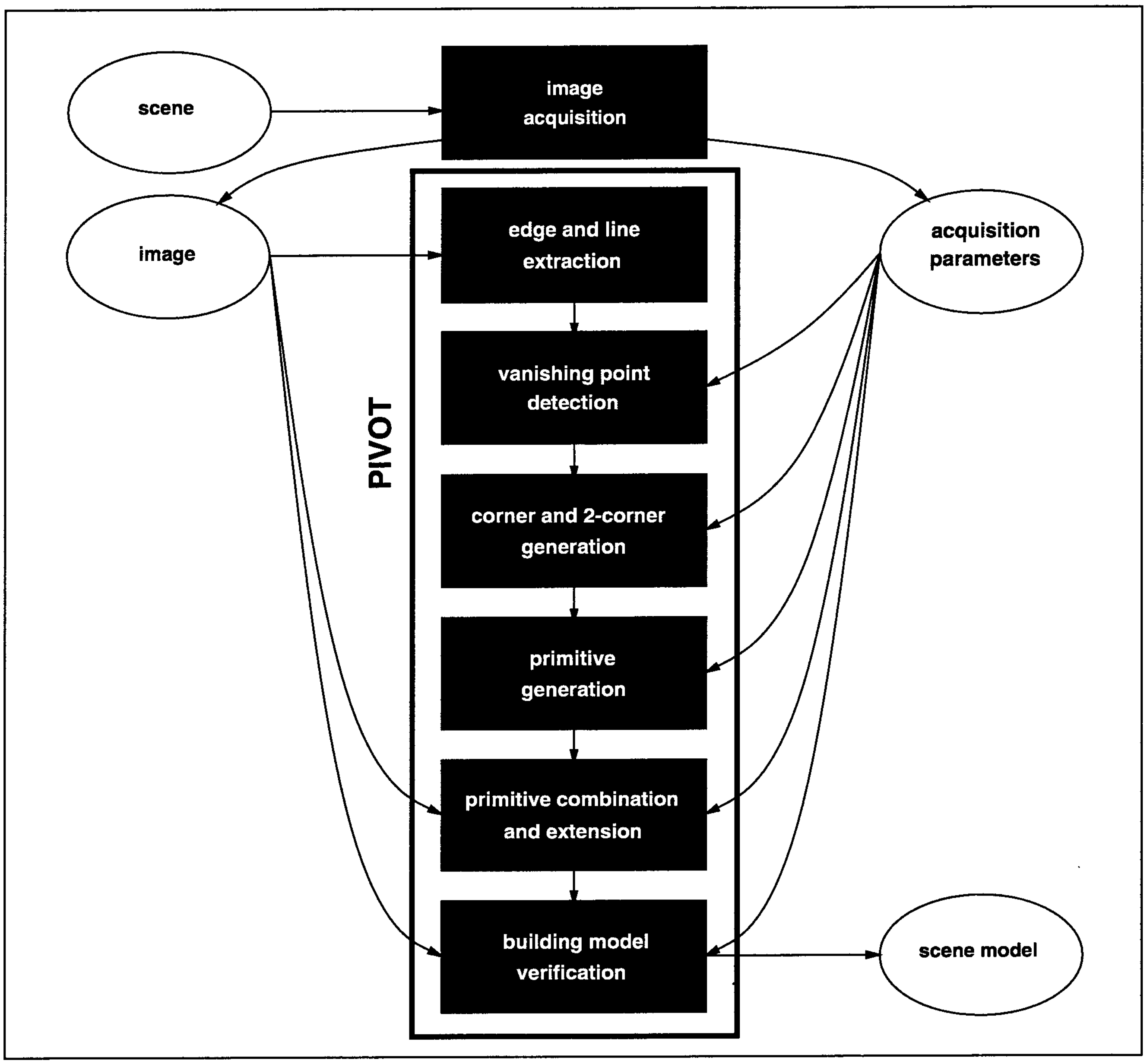

scene
image acquisition
image
acquisition parameters
edge and line extraction
vanishing point detection
corner and 2-corner generation
primitive generation
primitive combination and extension
building model verification
scene model
PIVOT

Figure 2-11: PIVOT process flow

PIVOT's top-level design represents one particular choice among several tradeoffs. In particular, PIVOT uses monocular analysis rather than stereo or multi-image analysis; PIVOT employs a central projection camera model, appropriate for frame cameras but not for other sensor types; PIVOT is data-driven, rather than model-driven (bottom-up, not top-down). Each of these design tradeoffs is discussed in turn.

PIVOT is a monocular system, meaning that it operates on single images, rather than on stereo or multiple view image sets. This design choice was made to explore performance in the limiting case: to what extent can 3D structure reliably be recovered from a *single* image with the use of a camera model? Since humans have little difficulty in manual reconstruction of 3D shape from a single view, it is natural to ask how well a fully-automated computer vision system would perform at the same task. Moreover, by limiting the analysis to single-view situations, the issues of correspondence and multi-image matching can be removed from consideration. These issues are still active open topics for research, particularly in the aerial image domain [18, 19, 81, 89].

Finally, it can be argued that while the perception of *depth* is inherently a stereo problem, the perception of *shape* is inherently a monocular one. The line-labeling work of Waltz [102] and the origami world of Kanade [48] strongly suggest that our perception of line drawings is essentially monocular in nature. To the extent that the scene is composed of regular polygonal shapes, it is reasonable to expect monocular analysis to bear the processing load.

As mentioned in Section 2.1, PIVOT assumes that the sensor is a standard frame mapping camera, which implies a central projection camera model and hence a perspective imaging geometry. An alternate class of sensors, typical of satellite-based remote sensing, are multispectral line array CCD scanners, or "pushbroom scanners," so named because the linear array "sweeps" the area to be sensed, acquiring data a line at a time. Here, the geometry is quite different, in that perspective projection holds along the line, while the inter-line geometry is orthographic. In this geometry, unlike perspective geometry, straight lines in the world do not always appear as straight lines in the image.

A key aspect of PIVOT's analysis, as will be seen in Chapter 3, is its reliance on the use of vanishing points to detect 3D structure. Vanishing points are a consequence of the perspective projection of parallel lines in object space; under non-perspective projection models such as pushbroom sensor models, different geometric techniques are required for analysis. While PIVOT currently operates exclusively under a central projection camera model, the use of generic photogrammetric methods will provide a basis for implementing analysis techniques for non-perspective imaging systems. In previous work, a photogrammetric building extractor [67] was used with both a frame camera model and the FBIP (**F**ast **B**lock **I**nterpolation **P**rojection) camera model, which approximates a pushbroom model by piecewise frame approximations. This was accomplished through the use of an object-oriented photogrammetric toolkit [65], which provided transparent access to basic geometric operations under these two different camera models. A similar approach will prove useful in future extensions of PIVOT to alternate sensor models.

PIVOT is a bottom-up system, beginning with raw edge and line data and incrementally creating increasingly detailed intermediate representations to reconstruct primitive models. This stands in contrast to a top-down approach, which begins by instantiating building models and then looks for support for these models, based on rules which encode scene structure. In aerial image analysis, one popular approach is to use a digitized map to guide the search for objects.

The difficulty with the top-down approach is its sensitivity to noisy data. When objects are not occluded, object boundaries are sharp with high contrast, and objects can be modeled well by the chosen representation, top-down methods can achieve good performance. However, top-down methods exhibit brittle performance when occlusions, edge fragmentation, or variable object shape and size are present. These factors all impede the ability to fit model instances to image data. Defining *a priori* rules for building models in aerial images is extremely difficult, given the wide variability of urban and suburban scenes [13].

Before closing this chapter and beginning a detailed discussion of the algorithms and techniques used by PIVOT, it is worthwhile to revisit the object detection and delineation principles outlined in Figure 2-10, in order to anticipate the degree to which current state-of-the-art techniques can follow them. Various existing vision systems have already followed some of these principles in different problem domains, yet have failed to exhibit robust and accurate performance. While the argument has been made that this set of principles describes necessary conditions for high-performance object detection and delineation, it is reasonable to wonder whether this set of principles is sufficient. Since no existing system has rigorously followed all of these principles, this question remains open.

Assuming that these principles are sufficient conditions for robust and accurate object detection and delineation, another natural question is whether some of these principles are harder to follow in practice than others. As Chapters 4 and 5 will illustrate, the fourth principle, which states that no interpretations should be discarded before their time, is difficult to implement without encountering combinatorial explosions in the number of legal interpretations of image data. The utility of geometric constraints derived from rigorous camera models will be clear in these chapters; it will be equally clear that more constraints are needed to produce a complete solution of the object detection and delineation problem.

Nonetheless, the principles outlined in this chapter represent a starting point for the development of such solutions. Grounding the discussion of object detection and delineation algorithms against these principles is an effective way to highlight strengths and weaknesses of the algorithms, and the results produced by PIVOT are clear evidence of the utility of these principles as guides for algorithm design. Chapter 3 begins the description of PIVOT's processing phases, with a thorough discussion of vanishing point analysis, the method by which Principle 1 is applied.

Chapter 3
Primitives and Vanishing Points

The discussion of Chapter 2 presented several principles for generic object detection and delineation. In particular, Principle 2 advocated the use of primitive volumetric forms as building blocks for complex object models. The idea behind such a strategy is that these simple primitive forms can be readily extracted from image data, while still maintaining expressive power by combination into complex shapes. This approach raises two natural questions:

- What primitives will be used?
- What constraints and features can be used to reliably extract them?

This chapter answers both of these questions. The beginning of the chapter focuses on a detailed description of the primitives to be used throughout this work, and the motivations behind this particular selection of primitives. The geometry of these primitives is of particular interest, since Principle 3 will be employed to prune the search space down to features which obey this geometry.

The remainder of the chapter turns to the second question, which is answered through the aid of rigorous camera modeling, in accordance with Principle 1. Under central projection, a set of parallel lines in the world maps to a set of lines in space which converge on a single point. Such a point is known as a *vanishing point*, and it can be applied to label line segments in the image with likely orientations in object space.

Methods for vanishing point detection are developed which make use of the geometry of primitives to guide the detection process, and two distinct models of edge uncertainty are presented to account for error in the detection process. The chapter concludes with a thorough experimental analysis in which these methods are shown to provide robust performance [91].

The basic mathematics of coordinate systems, vanishing points, and the Gaussian sphere representation of 3-space orientation is described in Appendix A. The reader who is unfamiliar with any of these topics is encouraged to read the appendix first before proceeding, as much of this chapter and succeeding chapters rely heavily on these results.

3.1. Selecting primitives

As discussed in Section 1.2, a number of primitive objects have been employed for object modeling. Generalized cylinders, superquadrics, physically-based models, and geons have all seen use in a variety of vision domains. As discussed earlier, a common problem with many of these representations is that they can be difficult to extract from real image data without the use of range data, user intervention, or strong initial conditions.

The goals of a particular choice of primitives are threefold. First, they should be well-suited for the problem domain; second, the chosen primitives should lend themselves to extraction from complex imagery, even in the face of the multitude of effects which make generic object detection and delineation difficult; third, they should combine to span most, if not all, of the object "language" in the domain. Each of these goals is discussed in turn.

It is not a surprising observation that the choice of primitives should be motivated by the problem domain. Systems which are meant for natural scenes face an entirely different representational problem than those for manmade environments. For example, the fractal models which might be appropriate for representing natural features such as trees and mountains are inappropriate for modeling the regular, well-defined shapes which characterize objects in manmade scenes. In aerial images, the dominant shape is the rectangular volume, which demands a simple model.

A particular choice of primitives should also be based on the degree to which those primitives can be readily extracted from the available data. Primitives which require range data, user intervention, or strict initial conditions (to name a few restrictions) limit the applicability of the system. In this sense, the ideal choice for a set of primitives would be those which can be derived from a single intensity image with the use of a rigorous camera model over a wide variety of scenes, and nothing else.

Another important aspect of this choice is its overall representational power. The chosen set of primitives ought to combine to span the possible set of objects in the domain, much as a limited set of phonemes can be combined to span a much larger set, the set of possible words for spoken language. The set of primitives should be just large enough that its combinations represent all expected object shapes in the domain of interest, but not so large that an undue burden is placed on the detection process.

The choice of primitives made in this work, specific to the aerial image domain, satisfies all of these goals. The two primitives to be described shortly are well-suited for modeling a variety of buildings that appear in aerial images; they can be extracted from complex images using vanishing point geometry (as we will see in this chapter and others); and they combine to represent a large number of manmade structures. The next section details the particular primitives to be used in this work.

3.2. Rectangular and triangular volumes

In the aerial image domain, it is generally the case that manmade structures sit on the ground. Mathematically, this means that our primitives will lie on planes in object space which are parallel to the *XY*-plane. Care should be taken in reading this statement; it should not be interpreted as meaning that the ground is necessarily planar. It should be interpreted as meaning that the floors of manmade structure are perpendicular to vertical lines in object space, which does not preclude local variations in terrain. Buildings in hilly and mountainous regions can be modeled, assuming their floors are flat. This observation effectively removes one degree of freedom from our primitive models (by fixing the elevation angle), limiting the hypothesis space by Principle 3 without violating Principle 4.

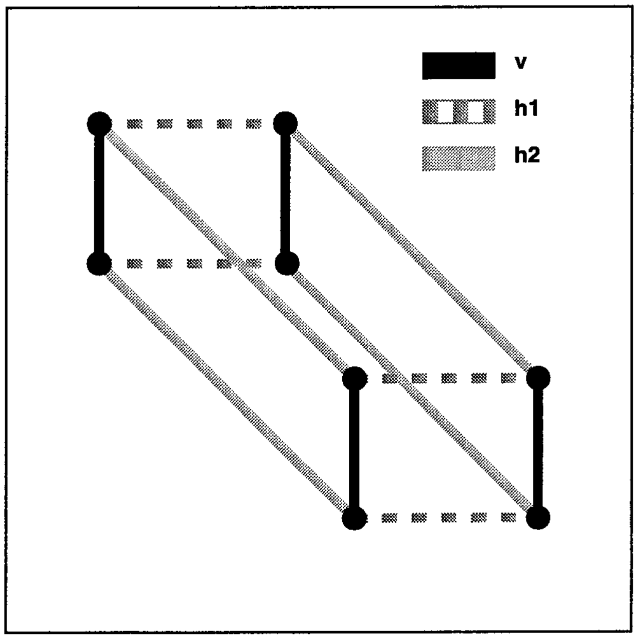


Figure 3-1: Rectangular volume primitive

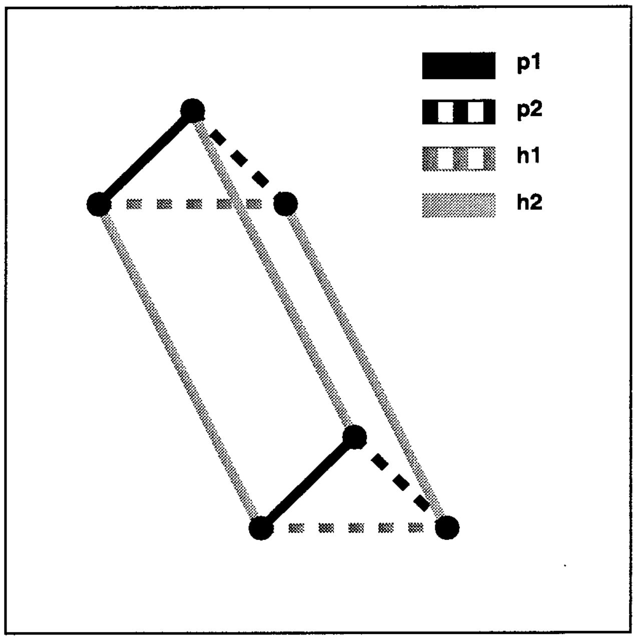


Figure 3-2: Triangular volume primitive

The first of the two primitives to be used throughout this work is the rectangular volume, shown in Figure 3-1. The edges of this primitive have three possible orientations; each of these orientations is orthogonal to the other two. One of these orientations (v in the figure) is parallel to the object space *Z*-axis; the other two (h1 and h2) are parallel to the *XY*-plane, but need not be parallel with the *X*-axis or *Y*-axis (which allows rotations around vertical axes).

The second primitive is a triangular volume, shown in Figure 3-2. It is a prism formed by three rectangles, joined at the ends by two congruent isosceles triangles which lie in parallel vertical planes. The base of the prism is parallel to the *XY*-plane, and the triangles point in the $+Z$ direction, away from the ground. The extrusion edges (h2 in the figure) are orthogonal to the triangle bases (h1). The peak edges can have one of two orientations (p1 and p2); the cross product of these orientation vectors is parallel to h2.

Of course, these two primitives do not span the entire set of manmade structures; n-sided pyramids and gabled rooftops cannot be modeled well with these volumes, nor can curved structures such as cylinders, spheres, and domes. Nonetheless, the combination of these primitives generates sufficient representational power to model a substantial portion of manmade structures, including buildings comprised of multiple rectangular wings such as L, T, and E-shaped structures, as well as split-level rectangular buildings. This particular set also has the advantage of being the smallest set which still permits useful exploration of the issues involved in primitive combination, due to the prevalence of buildings comprised of rectangular and triangular volumes.

The geometry of these primitives is of particular importance, as it will guide the search for primitive hypotheses. Equally important is the observation, to be taken up in Section 3.4, that the geometric cues needed to generate these hypotheses fall in the class of cues which are intrinsic to the imaging process, as Principle 1 mandates. The next section discusses vanishing points, the basic mechanisms for primitive generation.

3.3. Previous methods for vanishing point detection

Under central projection, a set of parallel lines in the world maps to a set to lines in image space which converge on a single point in the image. This point, known as a *vanishing point*, is an implicit representation of the 3-space orientation of the parallel lines in the world, as the mathematics of Appendix A illustrate. Given a vanishing point **v**, linear features in an image which appear to pass through **v** can be hypothesized to have the 3-space orientation represented by **v**. This approach allows a system to assign a few plausible 3-space orientations to linear features without having to consider all possible orientations defined by the interpretation planes of the features.

The automatic detection of vanishing points is based on two general assumptions about the nature of the scene; that sets of parallel line segments exist in the scene, and that these line segments in the scene project to line segments in the image. To the extent that these properties are obeyed, vanishing points can be readily extracted from an image. Several approaches have been proposed for the detection of vanishing points in an image; a survey can be found in [98]. Many of these methods are variations or enhancements of the basic ideas proposed by Barnard.

Barnard [4] divided the problem into two steps; finding line segments in the image, and finding intersections of the lines formed by these segments which constituted likely vanishing points. The first step was achieved by conventional edge detection techniques. The second step was achieved by computing the interpretation planes of these line segments, computing the great circles corresponding to these interpretation planes, and then tracing these circles in a two-dimensional array representation of the Gaussian sphere indexed by azimuth and elevation. The tracing was done by using Equations (A.10) and (A.11). The array served as a histogram; where many circles intersected, a peak was formed in the histogram. This peak corresponded to a vanishing point on the Gaussian sphere, which could be computed directly from the azimuth-elevation indices of the histogram peak by Equation (A.7).

This method, which can be described alternatively as a Hough transform on azimuth-elevation parameter space, has several known drawbacks: vanishing point accuracy is only as good as the quantization of parameter space; vanishing points lying near cell boundaries may have their votes split among multiple cells; uniform discrete quantization in azimuth and elevation leads to non-uniform cell sizes in the histogram; and bias and noise can be introduced by the Hough transform itself.

Many methods have been proposed to address these shortcomings. Some methods are variations on Barnard's method; others use essentially different techniques to detect vanishing points. The following description of existing approaches is not comprehensive, but is representative of the spectrum of vanishing point detection techniques and their drawbacks.

Magee and Aggarwal [63] employed cross product operations to compute the intersections of pairs of line segments directly. Vanishing points were then found by locating clusters of these intersection points, maintained as lists rather than histogram cells. While such an approach potentially allows greater accuracy in vanishing point estimation, this process required pairwise comparison of all line segments, an $O(n^2)$ process.

Quan and Mohr [88] used a hierarchical Hough transform based on a pyramidal recursive subdivision of the Gaussian sphere. It is argued that this recursive subdivision of parameter space (essentially a quad-tree generation scheme) leads to efficient vanishing point detection in the average case; however, no complexity analysis was performed to evaluate this claim. This method also inherits many of the drawbacks of the Hough transform.

Collins and Weiss [20] estimated vanishing point location as the polar axis of an equatorial distribution of interpretation plane normals on the Gaussian sphere, allowing a statistical determination of the estimate's error. This approach is based on the observations behind Equation (A.16), and is equivalent to a least-squares plane fit of the interpretation plane normals. However, this approach assumed that image line segments were already clustered into groups of convergent lines.

Brillault-O'Mahony [12] modeled uncertainty in image acquisition, digitization, and edge detection by assuming that edge endpoints obeyed the Gaussian law. This resulted in an isotropic accumulator space where the probability of detecting a vanishing point was uniform across the space. This method, another variation of the Hough, also inherited many of its drawbacks.

Lutton, Maitre, and Lopez-Krahe [61] employed a two-stage Hough transform to find orthogonal vanishing point triplets. The first stage used Barnard's method with a semi-regular quantization of the Gaussian sphere to find a set of potential vanishing points, and the second stage performed a second Hough transform to locate orthogonal triplets. The technique also incorporates corrections for bias due to finite image extent [15, 64].

McLean and Kotturi [73] used a histogram quantization of image gradient orientations to hypothesize a set of potential vanishing points, and then performed clustering of lines to these hypotheses to refine vanishing point estimates, using covariance information on lines to guide the clustering. However, the initial set of hypotheses were placed on a circle in the image plane, centered at the image origin with fixed radius; as a consequence, multiple vanishing points lying on a radial line from the image origin would be difficult to distinguish, and vanishing points which are near the principal point would be split among all of the initial vanishing points prior to clustering.

In related work, Kanatani [50, 51] presented detailed error analyses for interpretation plane normals and vanishing points, using statistical models of image noise. However, these models were based on a *small image approximation*, which assumed that the distances from image features to the principal point were small relative to the focal length of the camera. Weiss, Nakatani, and Riseman [104] gave an analysis of vanishing point and vanishing line error, for the application of determining surface orientation by using two independent vanishing lines. They assumed, however, that the error for a vanishing point is a circular disk on the sphere, rather than utilize a more general covariance ellipse model.

Although this summary is by no means complete, it gives a representative picture of the types of approaches used to detect vanishing points and model uncertainty in the detection process. Methods based on the Hough transform are the most popular, although they have the drawbacks enumerated earlier. Other approaches which avoid the Hough have their own unique problems, such as asymptotically inferior runtime, or requirements that parallel lines be grouped beforehand.

For this work, the Hough transform will be used as the basis for the new developments in vanishing point detection. Despite its known drawbacks, it also offers certain advantages: it has linear time complexity, its strengths and weaknesses have been thoroughly documented in the literature, and its use affords the opportunity to empirically validate and/or refute existing Hough-based techniques.

3.4. Primitive-based vanishing point detection

The combination of object models with the Gaussian sphere is not a new idea. Horaud [38] developed a model-based recognition scheme which performed matching in 3D, using backprojection (described in Section A.4) to translate image features into 3-space. The Gaussian sphere has also been used as the basis for representations of polyhedral and curved objects. Horn provides a classic summary of one such representation, extended Gaussian images [39]; Kang and Ikeuchi have recently developed extensions to this representation [52]. These methods, however, are best suited to model-based recognition approaches, where both object shape and extent are explicitly known.

Given any of the vanishing point detection tools in the previous section, one might attempt to initiate image analysis by using these tools straightaway to find vanishing points, followed by tests to determine which line segments pass through particular vanishing points. While such an approach makes good use of the imaging geometry, it neglects the primitive geometry. A better approach would guide the search for vanishing points by using the geometry of the primitive volumes, just as the model-based approaches use the geometry of the object models. This section describes a new approach for using knowledge about object shape to guide the vanishing point detection process.

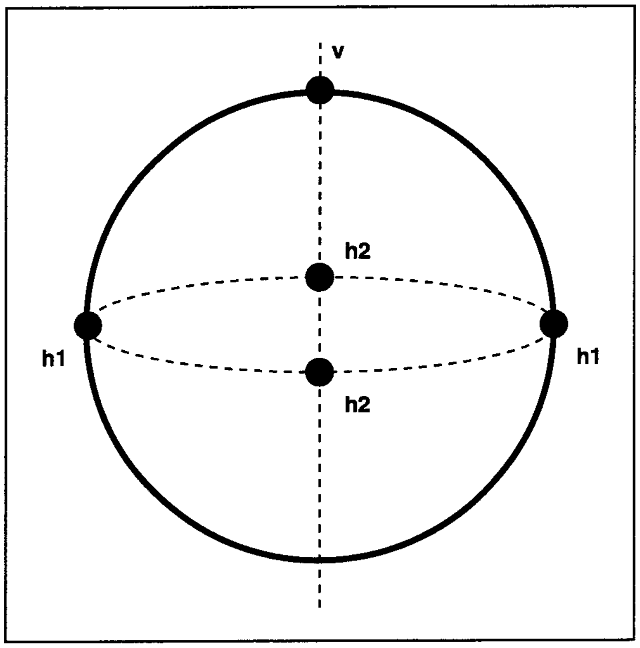


Figure 3-3: Orientations of a rectangular volume

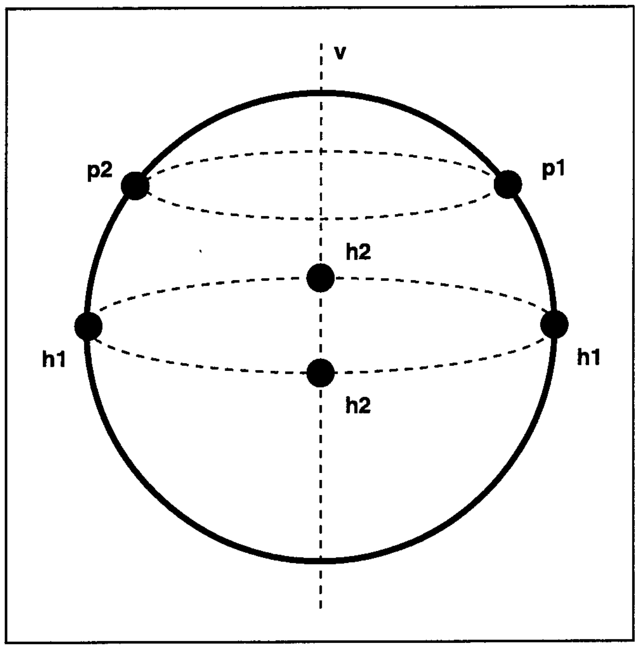


Figure 3-4: Orientations of a triangular volume

Equation (A.13) in Appendix A can be easily used to compute the vanishing point of each edge of a primitive in object space, given an image orientation matrix. This matrix is typically produced by an image resection, which determines the exterior orientation parameters of the camera; namely, the position of the optical center and the direction of the optical axis. Given the orientation of the image coordinate system, a rotation matrix can be computed which describes the rotation between the world and camera coordinate systems. This matrix, **M**, is the image orientation matrix. Throughout this work, **M** is assumed to be given; details of the computation of **M** can be found in Section A.1.

If Equation (A.13) is used to compute the vanishing point for each edge of the primitives depicted in Figures 3-1 and 3-2 in Section 3.2, the vanishing point pattern for each primitive can be obtained. Figures 3-3 and 3-4 show the relative vanishing point locations for the rectangular and triangular volumes, respectively. In each figure, the vanishing points have been labeled to correspond to the orientations they denote.

In both figures, **v** denotes the vertical vanishing point, although it should be noted that the triangular prism has no vertical lines. Each primitive has two orthogonal horizontal vanishing points, **h1** and **h2**. In addition, the triangular prism has two pairs of parallel slanted lines, which correspond to the vanishing points **p1** and **p2**. Since the triangular facets of the prism lie in vertical planes, the vanishing points for the slanted edges also lie in a vertical plane.

A hierarchical vanishing point structure can be observed in these diagrams. If the vertical vanishing point is known, then the horizontals for a particular primitive instance lie somewhere on the equator of the Gaussian sphere. If the horizontals are known, then the vanishing points for the slanted lines of a triangular volume must lie in one of the two great circles which pass through a horizontal vanishing point and the vertical vanishing point. Further, these two slant vanishing points must lie on the same horizontal plane slicing through the sphere.

This structure highlights certain symmetries of the primitive volumes and the vanishing points they generate. The rectangular volume has three planes of bilateral symmetry and three axes of rotational symmetry; the triangular volume has two planes of bilateral symmetry and one axis of rotational symmetry. These symmetries are reflected in the vanishing point representation on the Gaussian sphere. Bilateral symmetries of primitive volumes correspond to bilateral symmetries of vanishing points with respect to planes passing through the center of the Gaussian sphere, and rotational symmetries of the volumes correspond to rotational symmetries of vanishing points about axes passing through the center of the sphere.

The hierarchical and symmetric structure of the vanishing point sets for each primitive can be exploited to guide the search for vanishing points on the Gaussian sphere. Instead of scanning the sphere to look for maxima corresponding to individual histogram buckets, the essential idea is to scan the sphere and sample the histogram at multiple buckets, with sampling patterns based on the vanishing point representations depicted in Figures 3-3 and 3-4.

The technique is implemented as follows. First, the Nevatia-Babu [78] edge operator and linker is used to produce edges, which are broken into straight edge segments by recursive line fitting. The interpretation plane for each edge segment is then computed by a plane-fitting procedure and stored as an attribute for each segment. The Gaussian sphere is then used to histogram the great circles formed by these interpretation planes, as in Barnard [4].

Next, the vertical vanishing point is calculated directly from the image acquisition information with Equation (A.13), setting $\mathbf{q} = [0, 0, 1]^T$ as in Equation (A.14). This avoids the potentially noisy search for the vertical vanishing point on the Gaussian sphere histogram, which can fail outright in nadir and near-nadir views when vertical line segments in the image are short or not visible.

Next, scans are made to locate orthogonal pairs of horizontal vanishing points. This is done by sampling the great circle defined by the vertical vanishing point (alternatively, the equator with respect to the vertical axis). The circle is sampled in pairs, at ι and $\iota+\pi/2$, sweeping ι through an angle of $\pi/2$. At each value of ι, the bucket values at ι and $\iota+\pi/2$ are summed. The maxima of these pairwise sums then corresponds to a pair of horizontal vanishing points.

The horizontal vanishing points are then converted to their image plane representations, and each line segment is tested to see whether its extended line passes through one of the points. If it does, the great circle corresponding to the line segment is subtracted from the histogram. This entire horizontal vanishing point detection process is repeated until the largest pairwise sum is smaller than the largest bucket value found on the entire sphere. This strategy allows multiple pairs of horizontal vanishing points in scenes where rectangular structures do not lie on a uniform grid.

Once horizontal vanishing point detection is completed, a similar process is initiated to find slant vanishing points (e.g., **p1** and **p2** in Figure 3-4). Each pair of horizontal vanishing points defines two vertical planes; each plane is defined by a horizontal vanishing point, the vertical vanishing point, and the origin. Each plane defines a great circle on the sphere. Each circle is sampled in pairs, at ι and at its reflection through the other plane, sweeping ι through an angle of π/2 (from the equator to the vertical vanishing point). The maxima of these pairwise sums then corresponds to a pair of slant vanishing points. As before, the process is repeated, erasing great circles for line segments which pass through the vanishing points, and halting when the largest maxima is smaller than the largest sphere-wide maxima.

At the conclusion of this process, a set of vanishing points has been obtained, and each line segment has a set of attributes indicating which vanishing points it passes through. These can be treated as 3-space orientation estimates for each line segment. Related vanishing points are also linked; each horizontal vanishing point has a pointer to its orthogonal partner, and each slant vanishing point has a pointer to its partner. Also, each pair of slant vanishing points also has pointers to the horizontal vanishing points which define the symmetry planes for the slant vanishing points. This information is useful in corner generation, to form geometrically legal corners.

It is worth noting that the method described here could be generalized to arbitrary wireframe models or primitives. For any such primitive, the vanishing points could be computed for each line orientation in the primitive, and the resulting vanishing point structure could be used to guide the search for vanishing points on the Gaussian sphere, analogous to the search techniques described earlier. This point will be discussed again in Section 3.7.

3.5. Edge error modeling

A practical difficulty with the classical Gaussian sphere histogramming approach is its susceptibility to noise. Since every edge in an image will have its corresponding great circle added to the histogram, the possibility for spurious maxima increases with every image edge that is not associated with an object of interest in the scene. The problem is highlighted in building extraction tasks for aerial imagery, particularly in suburban areas where natural terrain and vegetation lie in close proximity with manmade structures.

Figure 3-5 shows one image of a stereo pair distributed as part of a test set on image understanding [28]. The scene contains several houses separated by streets, grassy yards, and scattered trees. At first glance, it would appear that this scene would readily lend itself to a vanishing point histogramming technique, since the houses and roads lie along orthogonal horizontal lines, and there appear to be no other dominant orientations in the image. However, when Barnard's method is applied to FLAT_L, the resulting horizontal vanishing points do not match the expected orientations. Figure 3-6 shows pairs of vectors indicating the directions of the computed horizontal vanishing points at regularly spaced points in the image. Obviously, these do not match the building and road horizontals.

This unexpected result can be explained by examining the contributions of the edge data to the histogram. Figure 3-7 shows two sets of edges; the white edges are those contributing to the two histogram buckets corresponding to the dominant horizontal peaks, and the black edges are those contributing to the buckets corresponding to the building horizontals. As can be seen, the white edges, generated by texture in the grassy areas of the image, will make a larger contribution to the histogram.

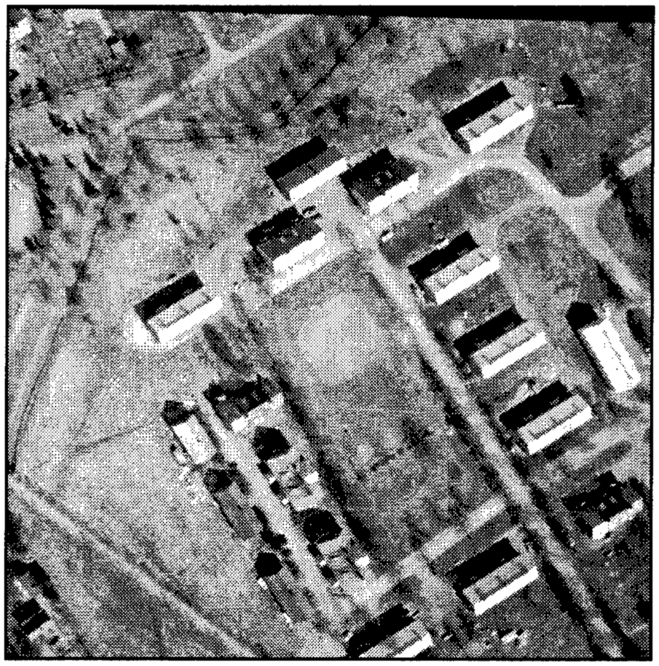


Figure 3-5: FLAT_L image

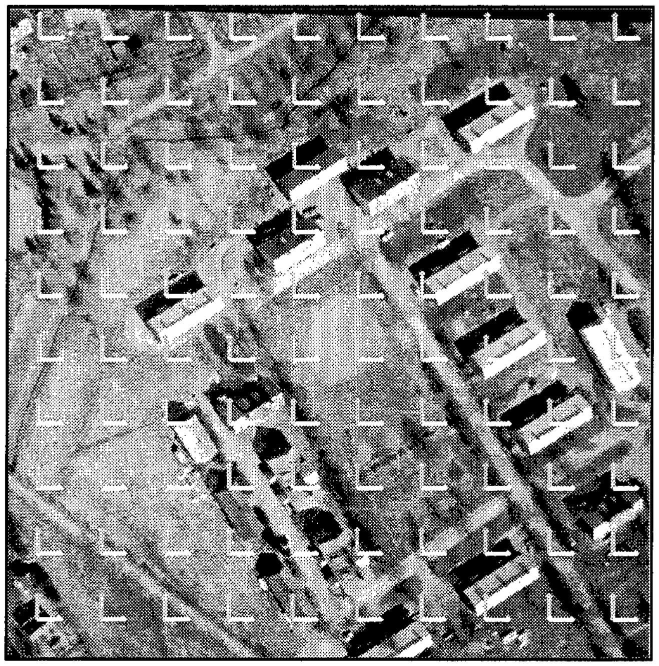


Figure 3-6: Computed dominant horizontals

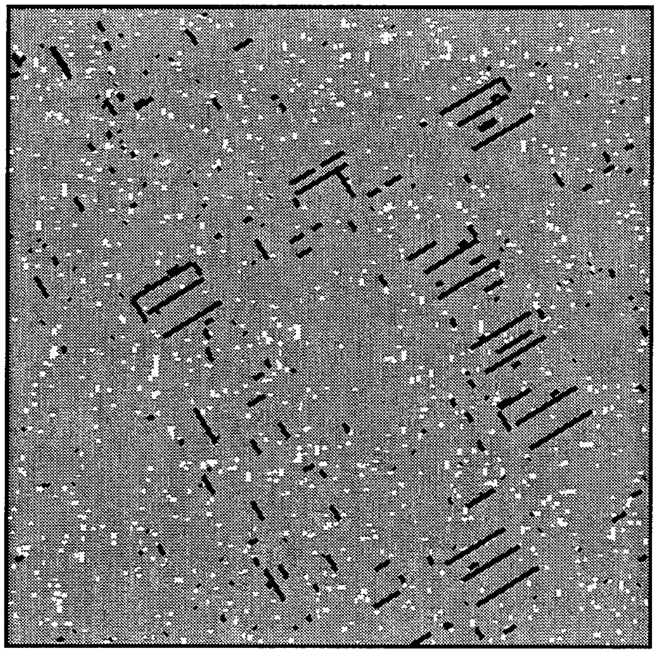


Figure 3-7: Edges contributing to buckets

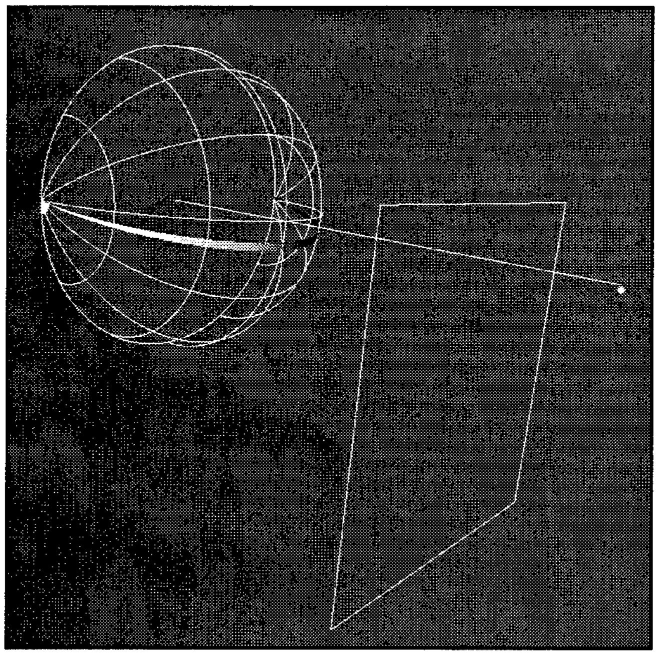


Figure 3-8: Histogram of one edge

It should also be noted that some building horizontals which appear to be missing from the figure actually contribute to buckets *adjacent* to the correct bucket in the histogram, further exacerbating the problem. Further, it is not an accident that the computed horizontals lie on the rows and columns of the image; given the discrete nature of edge operators and the short lengths of textural features in the image, short edges lying along rows and columns are a natural result.

Since our interest lies in the regular, straight lines that typify manmade structures (and lead to robust vanishing point solutions), it is natural to consider approaches which take these regularities into account during the histogramming process. Barnard used an unspecified heuristic taking line length and goodness-of-line-fit into account when adding values to the histogram [4]; the typical approach in the literature is to weight each great circle proportionally to the length of the line that generated it. While these heuristics can improve the vanishing point solution quality, they still fail when many "noise" edges are present, as the previous example illustrates.

Another solution is to threshold edges by length, filtering out short edges. While threshold-based filtering methods can be made to work for specific images by manual determination of the threshold, such methods fail to exhibit robust performance in the context of automated vision systems. It is often impossible to find a single threshold which works well across a variety of imagery, and in many cases, it is impossible to find a threshold for a single specific image. For example, in Figure 3-5, many of the slanted roof peaks produce edges with lengths similar to those of the texture edges. In this case, no reasonable threshold value is possible, since object edges will be discarded along with "noise" edges. Filtering constitutes a violation of Principle 4, since edges are being discarded which might have legal and correct interpretations as object edges.

An alternative idea is to model the orientation error in line segments and the interpretation planes they produce, given the discrete image geometry in which they are represented. To that end, two edge error modeling techniques are presented in the following sections. The first model allows the endpoints of a line segment to vary within their surrounding pixel boundaries, producing a swath of possible orientations for the segment. The second technique models the error on the interpretation plane normal with a Gaussian distribution.

Before proceeding with the discussion of the error models, we begin a running example which graphically illustrates the effects of these models. Figure 3-8 shows the Gaussian sphere histogram for one of these noise edges, a 4-pixel horizontal edge extracted from FLAT_L. Only the half-sphere facing the image plane is shown. The image bounds of FLAT_L are depicted as the rectangle; the line piercing the sphere is the optical axis. The dot at the end of the optical axis is the vertical vanishing point in the image plane. The vertical vanishing point lies just off of the optical axis, indicating slight image obliquity. The band running around the "equator" of the sphere is the set of buckets filled by the 4-pixel edge.

Intuitively, the width of the band indicates the degree of uncertainty in the orientation of an edge's interpretation plane. Without an edge error model, all interpretation planes are treated as having orientations accurate to within the bucket size of the histogram, independent of edge length or image position. The depiction in Figure 3-8 will be repeated in the subsequent discussion, using the same edge as an example, to illustrate the effects of edge error modeling.

3.5.1. Interpretation plane swath model

Recall that interpretation planes can be generated from line segments in the image plane by using three points; the endpoints of a line segment, and the perspective center. The *interpretation plane swath model* estimates the potential orientation error in an interpretation plane by allowing the two line segment endpoints to vary within their surrounding pixel boundaries.

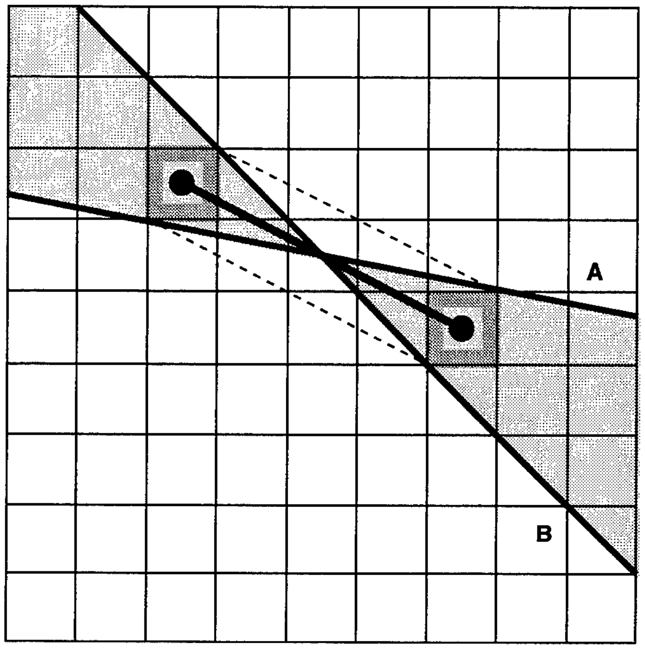


Figure 3-9: Interpretation plane swath

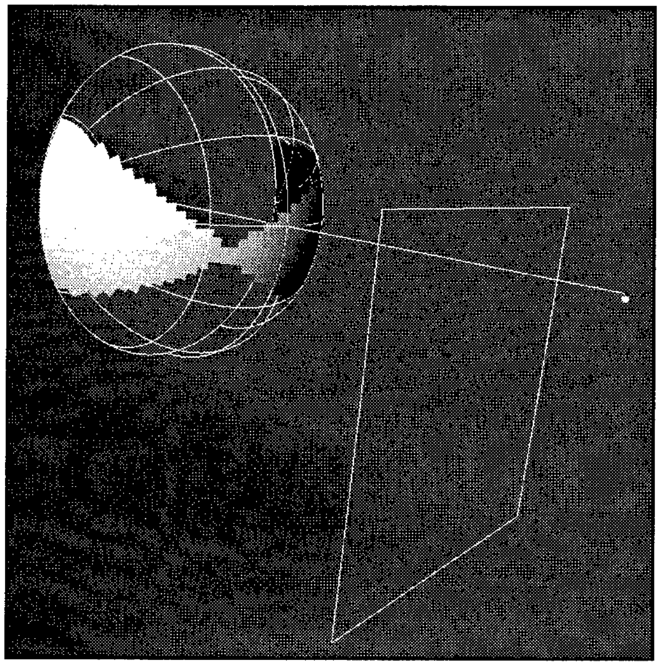


Figure 3-10: Swath example, one edge

This produces two extreme error bounds for the line segment, which can then be used to form bounding interpretation planes within which the true interpretation plane is expected to lie.

Figure 3-9 shows a line segment in an image. Its endpoints are defined by two pixels in the image. Since the positional accuracy of each endpoint is limited to the extent of its bounding pixel, there exists a range of possible positions for each endpoint. Intuitively, if we allow each endpoint to move within its bounding pixel, an area is swept out by the line connecting the endpoints. We can approximate the bounds of this area by two lines (A and B in the figure) which pass through extreme points of the bounding pixels. The approximated area is shown in gray; the dotted lines indicate the area neglected by this approximation. Intuitively, this model addresses uncertainty in line segment orientation, while neglecting positional uncertainty of the line segment.

The bounding lines A and B can then be used to form two interpretation planes, and hence two great circles on the Gaussian sphere. The area between these circles can then be histogrammed. The result of this process for our running example is shown in Figure 3-10. Here, the orientation uncertainty is represented as a swath on the Gaussian sphere. It should be clear that longer lines will generate narrower swaths on the sphere, and hence have less uncertainty in the orientation of their interpretation planes.

The advantage of this edge error model is that it is simple to implement and relatively efficient; each of the two great circles corresponding to A and B can be traced into the histogram, and then rows of buckets lying between A and B can be filled, not unlike a polygon boundary-fill algorithm. The disadvantages are that all possible interpretation plane orientations are treated equally (i.e., the swath has a uniform distribution); further, some orientation uncertainty is neglected, for the sake of simple approximation of edge error, as shown by the dotted lines in Figure 3-9. The model also assumes that uncertainty is limited to one pixel at each endpoint; due to factors such as line fitting to edge points, the amount of deviation in an edge may be greater or smaller, both at the endpoints and along the length of the edge. The next section considers a more rigorous edge error model.

3.5.2. Interpretation plane distribution model

While an interpretation plane can be generated by a line segment and the perspective center, as in the previous section, it can also be generated directly from an edge by a plane-fitting procedure, using the perspective center and the edge pixels for the plane fit. In this section, we consider the use of a least squares solution for the plane fit, which as a side effect produces a covariance matrix for the estimated plane parameters. The resulting confidence ellipse for the plane normal can be used to generate a Gaussian distribution for the great circle associated with each edge. Under this model, short edges lead to plane fits with large variances, producing a wide swath on the Gaussian sphere; long edges lead to smaller variances, and tighter swaths.

The idea of representing interpretation plane normal uncertainty by a covariance matrix is not new; Kanatani presented one such covariance model in great detail [50, 51]. However, this model is based on the *small image approximation*: given a point of interest at distance r from the principal point, and focal length f, assume that $r \ll f$. This assumption does not hold in general, and in particular is invalid for aerial mapping photography, acquired by cameras which typically have a focal length of 6 inches, with a 9×9-inch film format [94]. The model is also limited by the assumption that edge pixel density is isotropic with respect to orientation. The covariance matrix model presented here suffers from neither of these drawbacks; in addition, an algorithm is presented for translating the covariance information, expressed in terms of the camera coordinate system, to the azimuth-elevation parameterization of the Gaussian sphere.

A brief review of least squares plane fitting is in order. Assume that there are n data points for the plane fit, one of which will be the perspective center, and the remainder of which will be the points on the image plane which comprise the edge. Let (x_i, y_i, z_i) denote the i-th point. Let $\mathbf{B}$ be a $n\times3$ matrix, in which the ith row vector of the matrix is $\mathbf{b}_i = [x_i-\bar{x}, y_i-\bar{y}, z_i-\bar{z}]$, where $\bar{x}, \bar{y}$, and $\bar{z}$ are the means of the x_i, y_i, and z_i respectively. The scatter matrix for these points is then $\mathbf{S} = \mathbf{B}^{\mathrm{T}}\mathbf{B}$.

The squared error of the points with respect to a plane with normal $\mathbf{v}$ can then be defined as the sum of the squared lengths of the $\mathbf{b}_i$ along $\mathbf{v}$:

$$E^2 = \sum_i (\mathbf{b}_i \cdot \mathbf{v})^2 = \mathbf{v}\left(\sum_i \mathbf{b}_i^{\mathrm{T}} \cdot \mathbf{b}_i\right)\mathbf{v}^{\mathrm{T}} = \mathbf{v}\mathbf{S}\mathbf{v}^{\mathrm{T}} \tag{3.1}$$

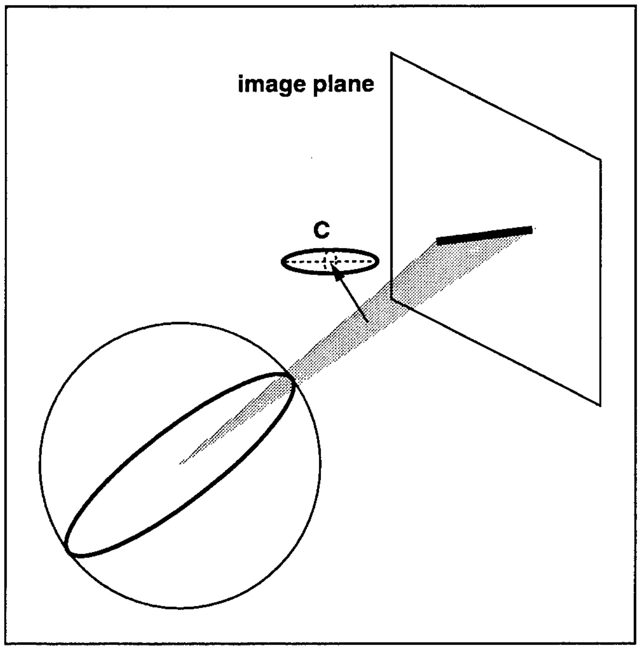


Figure 3-11: Plane fit error ellipsoid

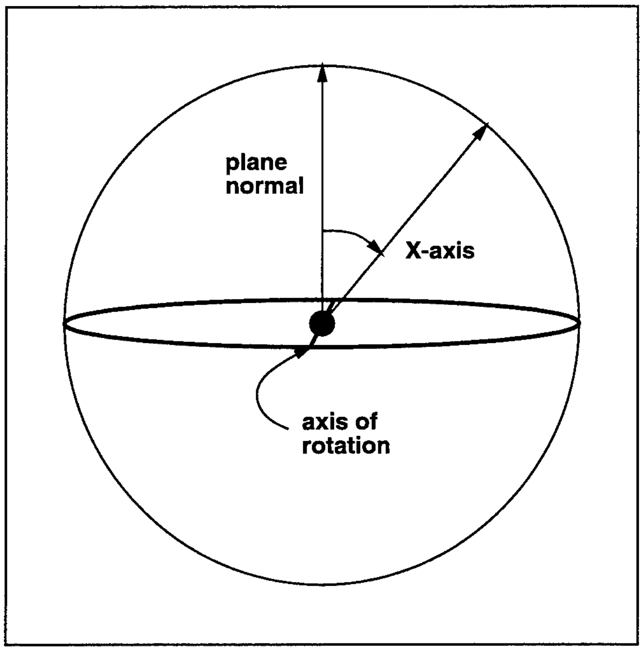


Figure 3-12: Rotating into the polar axis

E^2 can then be minimized by finding the $\mathbf{v}$ for which $\mathbf{v}\,\mathbf{S}\,\mathbf{v}^T$ reaches a minimum value; but this is simply the eigenvector corresponding to the minimum eigenvalue of $\mathbf{S}$. Given this eigenvector $\mathbf{v}=(\phi_x, \phi_y, \phi_z)$, the plane equation can be readily obtained by solving for d in $\phi_x\bar{x}+\phi_y\bar{y}+\phi_z\bar{z}+d=0$.

Since the plane equation is linear, the covariance matrix $\mathbf{C}$ for the direction cosines (ϕ_x, ϕ_y, ϕ_z) of the plane normal can be easily computed by the following equation, where σ_0^2 is the variance of unit weight [94]:

$$\mathbf{C} = \sigma_0^2 \mathbf{S}^{-1} \tag{3.2}$$

The diagonal entries of this matrix $(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ give the variances on the direction cosines. Figure 3-11 shows the geometric situation, where a plane has been fit to the perspective center and the points of an edge in the image plane. This interpretation plane slices through the Gaussian sphere, tracing a great circle. C is the error ellipsoid for the plane normal, derived from the covariance matrix $\mathbf{C}$. The essential idea is that $\mathbf{C}$ determines a Gaussian distribution for the great circle, where areas close to the great circle will be weighted more heavily than areas further away from it.

From an implementation standpoint, however, we seek variances in terms of the (α, β) azimuth-elevation quantization of the sphere. Direct expression of σ_α^2 and σ_β^2 in terms of σ_x^2, σ_y^2, and σ_z^2 can be accomplished by evaluating the following matrix equations:

$$\sigma_\alpha^2 = \left[\frac{\partial\alpha}{\partial\phi_x}\ \frac{\partial\alpha}{\partial\phi_y}\ \frac{\partial\alpha}{\partial\phi_z}\right] \mathbf{C} \left[\frac{\partial\alpha}{\partial\phi_x}\ \frac{\partial\alpha}{\partial\phi_y}\ \frac{\partial\alpha}{\partial\phi_z}\right]^T \tag{3.3}$$

$$\sigma_\beta^2 = \left[\frac{\partial\beta}{\partial\phi_x}\ \frac{\partial\beta}{\partial\phi_y}\ \frac{\partial\beta}{\partial\phi_z}\right] \mathbf{C} \left[\frac{\partial\beta}{\partial\phi_x}\ \frac{\partial\beta}{\partial\phi_y}\ \frac{\partial\beta}{\partial\phi_z}\right]^T \tag{3.4}$$

While the use of these equations is theoretically sound, difficulties arise in practice. Without loss of generality, assume that the polar axis of the sphere is coincident with the X-axis. It can then be shown that:

$$\frac{\partial\beta}{\partial\phi_x} = \frac{1}{\sqrt{1-\phi_x}} \tag{3.5}$$

$$\frac{\partial\alpha}{\partial\phi_y} = \frac{\phi_z}{\phi_y^2+\phi_z^2} \tag{3.6}$$

$$\frac{\partial\alpha}{\partial\phi_z} = \frac{-\phi_y}{\phi_y^2+\phi_z^2} \tag{3.7}$$

All three of these partials are infinite at the poles of the Gaussian sphere, which makes the use of Equations (3.3) and (3.4) problematic.

An alternate approach, and the one employed in this work, is to rotate the scatter matrix so that it aligns with the polar axis, effectively decorrelating the covariances of the direction cosines. Let $\mathbf{R}$ be the directed rotation matrix which rotates around the vector perpendicular to the interpretation plane normal and the X-axis, by an angle of $\cos^{-1}\phi_x$. Figure 3-12 illustrates the desired rotation. Then the rotated scatter matrix $\mathbf{S}_R$ can be computed by rotating $\mathbf{B}$ (denoted by $\mathbf{B}_R$):

$$\mathbf{S}_R = \mathbf{B}_R^{\mathrm{T}}\mathbf{B}_R = (\mathbf{BR}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{BR}^{\mathrm{T}}) = \mathbf{RB}^{\mathrm{T}}\mathbf{BR}^{\mathrm{T}} = \mathbf{RSR}^{\mathrm{T}} \tag{3.8}$$

The rotated covariance matrix is then:

$$\mathbf{C}_R = \sigma_0^2\mathbf{S}_R^{-1} \tag{3.9}$$

This leads to the following algorithm for histogramming a Gaussian-distributed great circle, given an interpretation plane ϕ (again assuming that the X-axis is the polar axis of the Gaussian sphere):

1. Compute the directed rotation matrix $\mathbf{R}$ for the interpretation plane.
2. Compute $\mathbf{C}_R$ using Equation (3.9), and obtain the diagonal entries σ_x^2, σ_y^2, and σ_z^2, where σ_x^2 should be zero due to decorrelation of the direction cosines.
3. For each bucket on the sphere, represented by a unit vector $\mathbf{b}$:
   a. Compute $\mathbf{b}_R = \mathbf{R}\mathbf{b}$. Let $\mathbf{b}_{Rx}$, $\mathbf{b}_{Ry}$, and $\mathbf{b}_{Rz}$ be the components of $\mathbf{b}_R$.
   b. Let D be the line in the yz-plane defined by the points $(0,0)$ and $(\mathbf{b}_{Ry},\mathbf{b}_{Rz})$.
   c. Compute the intersection point (p_y,p_z) of D with the standard error ellipse derived from $\mathbf{C}_R$, as depicted in Figure 3-13. Then let $\sigma = \sqrt{p_y^2+p_z^2}$. Intuitively, σ is the standard deviation of the Gaussian distribution at this particular azimuth of the great circle.
   d. Let $\sigma_\beta = \sin^{-1}\sigma$.

e. Let $k = \sin^{-1}\mathbf{b}_{Rx}$, and let $k_1 = k+h$ and $k_2 = k-h$, where h is half of the bucket size (i.e., if a bucket covers one degree in elevation, then h would be half of a degree).

f. Approximate the probability distribution of the bucket by a 1D Gaussian, using k_1 and k_2 to compute $0.5(\mathrm{Erf}(k_1/\sqrt{2}\sigma_\beta) - \mathrm{Erf}(k_2/\sqrt{2}\sigma_\beta))$, which is the area of the Gaussian between the bucket boundaries k_1 and k_2. Increment the bucket by this value.

In practice, steps 3a-3f are not performed on every bucket of the sphere, as the contributions of the Gaussian distribution taper off rapidly as the spherical distance from the great circle increases. Instead, $\sigma_{max} = \max(\sigma_y, \sigma_z)$ is used to generate bounding circles $\pm 3\sigma_{max}$ away from the great circle, and the boundary-fill algorithm of Section 3.5.1 is employed to touch only those buckets lying between the $\pm 3\sigma_{max}$ circles.

The result of this algorithm after execution on our running example edge is shown in Figure 3-14. The brighter areas of the band correspond to buckets which are closer to the mean of the distribution, and hence get greater weight; the darker areas correspond to the tails of the distribution. In this case, the uncertainty of the 4-pixel edge is spread over a large portion of the sphere; longer edges have lower variances, leading to tight narrow bands on the sphere. The advantage of this edge error model is that it rigorously accounts for orientation uncertainty in the interpretation plane, using least squares solution techniques. Its primary disadvantage is speed; as one might expect, the inner loop of the preceding algorithm (step 3) is computationally intensive. Significant speedups can be gained by unrolling the matrix multiplication and using lookup tables for the sine and error functions.

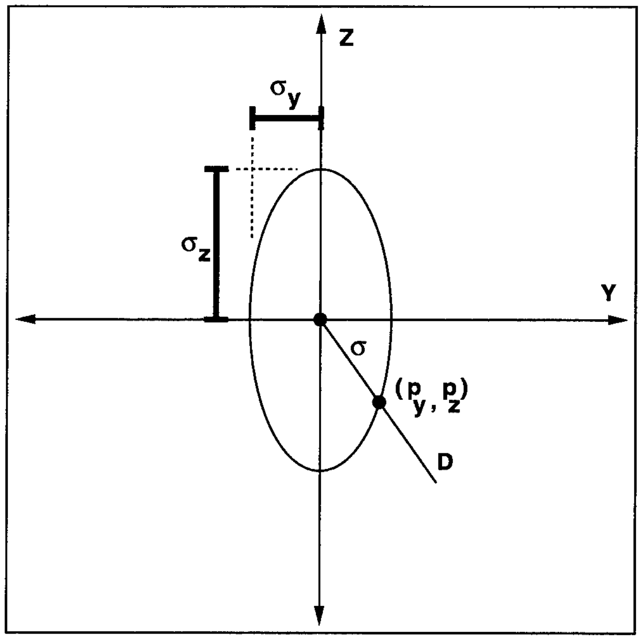


Figure 3-13: Computing σ by intersection

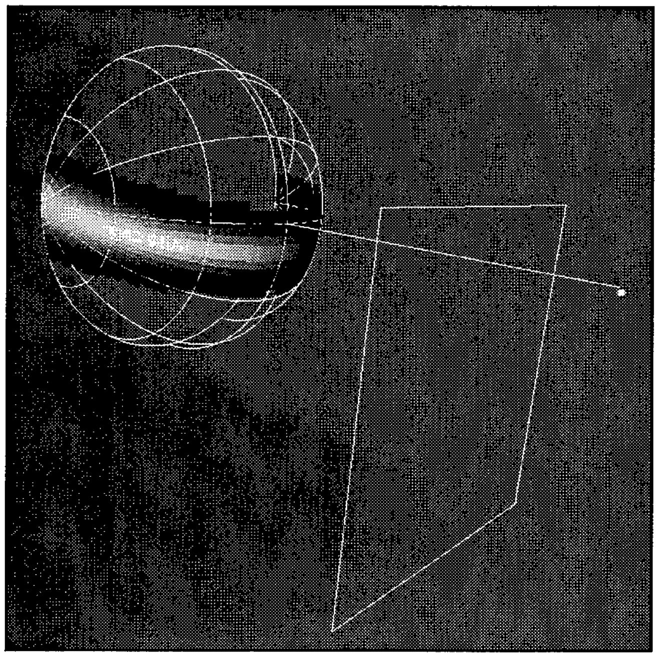


Figure 3-14: Distribution example, one edge

3.6. Performance evaluation and analysis

A vanishing point represents a 3D orientation in object space; more precisely, a vanishing point on the Gaussian sphere, represented by a unit vector **v**, maps to the object space orientation $\mathbf{M}^T\mathbf{v}$, where **M** is the image orientation matrix. Since the goal of most vanishing point applications is to find 3D orientations of specific structures in the scene, it is natural to ask how well the orientations of interest in the scene are approximated by automatically generated orientation estimates.

A natural performance metric is orientation error; given a particular orientation vector **p** in the scene, compute its error with respect to a particular vector **w** which has been automatically generated. The angular difference, or vector angle, between **p** and **w** is used to measure orientation error. Given a set of automatically generated vanishing points $\mathbf{v}_i$, we can trivially convert these to object space orientations by $\mathbf{w}_i = \mathbf{M}^T\mathbf{v}_i$. Then, for each true **p** in the scene, we can compute the smallest angular error with respect to the $\mathbf{w}_i$.

Such a comparison requires an accurate model of the scene, to serve as *ground-truth*. To generate scene models for performance evaluation, we used the SITECITY semi-automated modeling system [40, 41]. This modeler combines photogrammetric sensor models and integrated geometric constraints [66] with computer vision techniques, for computer-assisted generation of accurate 3D building models using multiple images. These models are represented as 3D wireframes in object space, which allows the orientations of specific building edges to be readily obtained.

For the performance evaluation described in this chapter, we used 22 images covering 6 scenes. Five of the scenes were taken from four images of Fort Hood, Texas, distributed under the RADIUS program; half of these 20 images were acquired from directly overhead, while the other 10 were acquired from oblique angles. The remaining pair of images was distributed as part of a test on image understanding techniques organized by WG III/3 of ISPRS [28]. The Fort Hood scenes contain barracks, garages, and other buildings, surrounded by roads and parking lots; the ISPRS scene contains suburban buildings, connected by roads and surrounded by grassy fields and scattered trees.

3.6.1. Experimental variations

For performance evaluation, Barnard's method was used as the baseline vanishing point detection algorithm. Various enhancements to the algorithm were then toggled on or off, as well as in combination, to assess the effects of these enhancements on vanishing point detection performance. The experimental variations employed in these experiments were:

1. **Statistical estimation**: Collins and Weiss [20] proposed refinement of vanishing point orientation estimates by treating a vanishing point as the polar axis of an equatorial distribution on the Gaussian sphere, and applying methods from directional statistics to refine the estimate, given the group of line segments that generated it. This method is equivalent to a least-squares plane fit of the heads of the interpretation plane normal vectors, using the resulting plane normal as the refined orientation estimate.

2. **Primitive-based detection**: this option, introduced in Section 3.4, searches for groups of vanishing points simultaneously, using known geometric relationships between them as defined by primitive shape models. In particular, the implementation used in these experiments searches for orthogonal horizontals, as depicted in Figure 3-3, and for vertically symmetric "peak" orientations, as depicted in Figure 3-4.
3. **Probability masking**: Lutton, Maitre, and Lopez-Krahe [61] observed that bias is introduced into the Gaussian sphere histogram by the finite extent of the image. Every great circle generated by a line segment will produce votes inside the projection of the image boundary on the sphere, which makes it more likely that a vanishing point will be detected inside the projected image boundary than outside of it. Their solution weights the histogram by a probability mask which takes this effect into account.
4. **Edge error modeling**: this option, introduced in Section 3.5, models the uncertainty in edge position and orientation, and transfers that uncertainty to the great circle histograms on the Gaussian sphere. There are actually two error models involved in these experiments: the swath model of Section 3.5.1, and the distribution model of Section 3.5.2. Each of these will be considered separately.

The first three of these options are binary in nature; either they are activated or disabled. The final option has three possibilities; either no edge error model is employed, the swath model of Section 3.5.1 is employed, or the distribution model of Section 3.5.2 is employed. In total, there are 24 combinations of experimental options available.

3.6.2. Experimental results and analysis

Each of the 24 experimental combinations described in the previous section was used on each of the 22 test images. For each of the resulting 528 test runs, a set of vanishing point hypotheses was generated. The ground-truth site model wireframes for the appropriate image were then broken into their constituent 3D line segments, each of which was treated as an orientation vector in object space and converted to a vanishing point by $\mathbf{v}'=\mathbf{M}\mathbf{v}$. Finally, for each ground-truth vanishing point, the angular error with respect to the closest hypothesized vanishing point was computed, and the average angular error for all ground-truth vanishing points was then calculated for that test run.

3.6.2.1. Statistical estimation

In the first experiment, we evaluated the effectiveness of statistical estimation for refinement of vanishing point orientation estimates. Since half of the 24 experimental combinations used statistical estimation, we can consider the change in average angular error for each test run when the statistical estimation process is toggled on and off, holding all other options fixed. This gives 22 images $\times$ 12 experimental variations, or 264 data points.

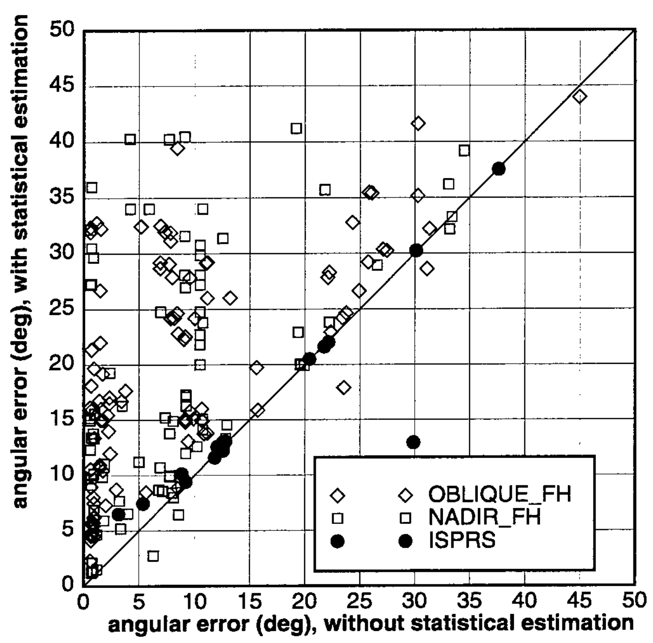


Figure 3-15: Effects of activating statistical estimation

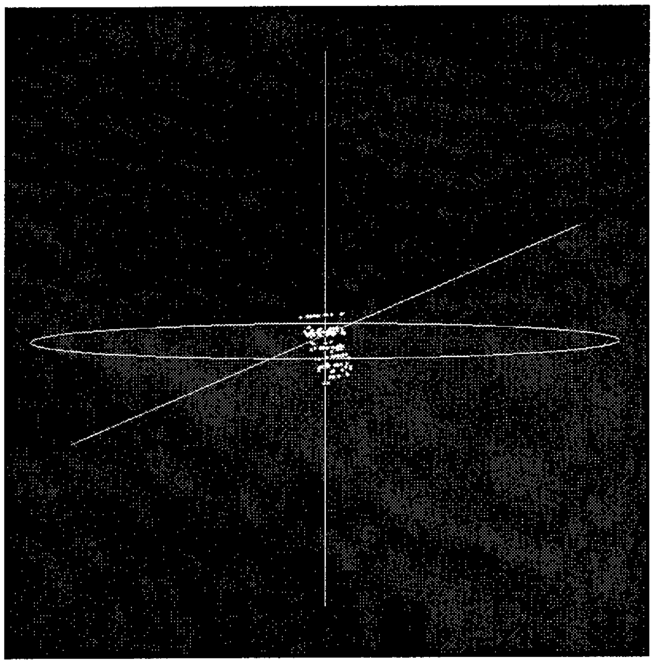


Figure 3-16: Example of estimation failure on a Fort Hood image

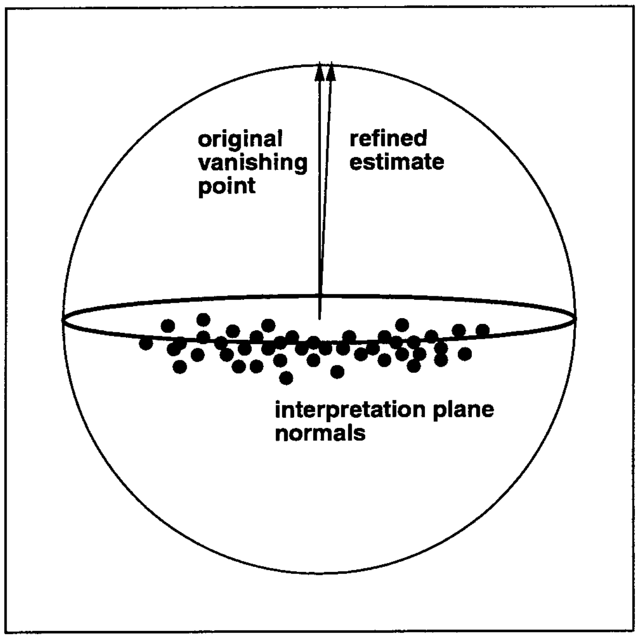


Figure 3-17: Strong image perspective effects

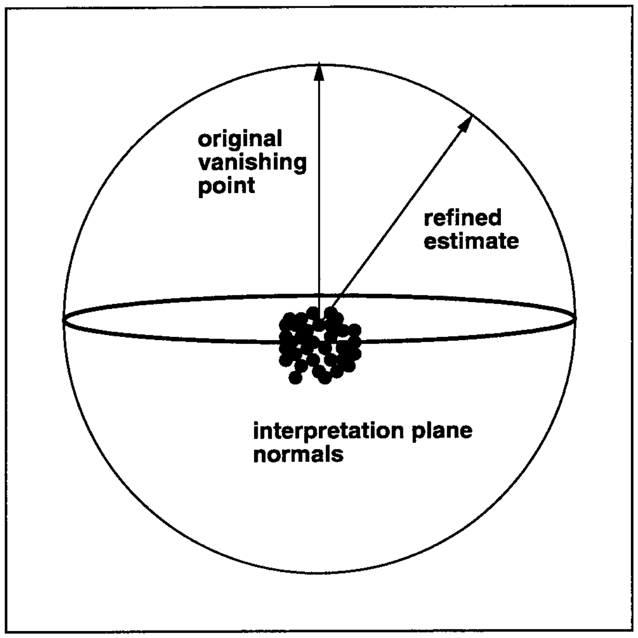


Figure 3-18: Weak image perspective effects

Figure 3-15 shows these data points, plotting the change in angular error as statistical estimation is toggled on and off. The data points are plotted in three groups; those representing tests on the ISPRS images, the nadir Fort Hood images, and the oblique Fort Hood images. The diagonal line represents no change in angular error. It follows that points lying to the lower right of the line represent an improvement in average angular error when statistical estimation is used; points lying to the upper left represent performance degradation when statistical estimation is used.

It is clear that the vast majority of the test runs show a significant decrease in performance when the statistical refinement process is invoked. In particular, many of the test runs on the Fort Hood images which originally exhibited an average angular error of less than 5 degrees show errors as high as 40 degrees when statistical estimation is invoked. With one exception, however, performance on the ISPRS images remained relatively stable when statistical estimation was invoked, becoming constant as average angular error increased.

Both of these results can be explained by examination of the data points used for statistical estimation. Recall that the statistical estimation process fits a plane to the heads of the interpretation plane normals for a particular vanishing point, using the resulting plane normal as the new vanishing point estimate. Figure 3-16 shows the geometric situation for a horizontal vanishing point automatically generated from one of the Fort Hood images. The line running from the top to the bottom of the figure is the true horizontal vanishing point, along with its corresponding great circle on the sphere. The diagonal line represents the refined vanishing point estimate, using the heads of the interpretation plane normals, shown as dots in the figure.

As the figure shows, the refined orientation estimate is far away from the original vanishing point, simply because the data points for the plane fit are closely packed and do not clearly lie in the desired plane. In geometric terms, this is due to the weak perspective effects of the Fort Hood images; line segments in the image, and hence interpretation plane normals, do not have a sufficiently wide spread to permit robust plane fitting. The contrast between strong perspective effects and weak perspective effects is graphically illustrated in Figures 3-17 and 3-18, where the spread of points with strong perspective effects leads to a robust fit, while the limited spread of data with weak perspective effects leads to a spurious fit.

The constancy of angular error for the ISPRS images can be explained by a similar argument. The texture edges for these scenes, illustrated earlier in Figure 3-7, consist of many short segments perfectly aligned with the rows and columns of the image. The resulting spread of interpretation plane normals lies perfectly on a great circle, leading to a robust plane fit in spite of the weak perspective effects. More succinctly, incorrect vanishing point solutions are consistently generated from noisy edges which happen to provide a robust plane fit. Of course, in absolute terms, the error is still large for many of the ISPRS test runs, since these texture edges lead to poor vanishing point solutions (as described in Section 3.5); statistical estimation cannot be faulted for fitting a poor orientation estimate to poor data. Nonetheless, the first experiment shows that the simple statistical estimation technique by plane fitting degrades solution quality for images with weak perspective effects.

Before moving to the next experiment, it is worth noting that simple numerical adjustments do not address the basic difficulty highlighted in Figures 3-17 and 3-18. A weighted least-squares plane fit could take into account line length, but there is no reason to expect that the points with greater weights will necessarily exhibit a better spread on the sphere. Alternatively, one might try to vary focal length to achieve a better spread, but this must necessarily spread all points, not just those which lead to the desired plane fit; the net result will be a spurious fit, as before. The nature of the difficulty is one of geometry, not of numerical stability.

3.6.2.2. Primitive-based detection

In the second experiment, we evaluated the effectiveness of primitive-based vanishing point detection. Eliminating statistical estimation from the set of experimental variations, 6 of the remaining 12 combinations used primitive-based vanishing point detection. Considering the change in average angular error as primitive-based detection is toggled on and off, holding other options fixed, we obtain 22 images × 6 experimental variations, or 132 data points. Figure 3-19 shows these data points, plotting the change in angular error as primitive-based detection is toggled on and off. As before, the data points are plotted in three groups according to the type of imagery, and the diagonal line represents no change in angular error.

As the figure shows, primitive-based detection improved average angular error in nearly all test runs, with large performance improvements for many of the Fort Hood scenes. This can be explained by noting that several of the Fort Hood images have only one dominant horizontal orientation, while the orthogonal horizontal orientation is much weaker. Figure 3-20 shows one of the Fort Hood test images, with the results of edge detection superimposed on the image. There are many segments corresponding to the horizontal orientation running from left to right across the image, but few corresponding to the orthogonal horizontal orientation. As a result, without primitive-based detection, only the dominant horizontal is reliably obtained, and the weaker horizontal is approximated poorly. Finding horizontals in orthogonal pairs significantly improves average angular error by explicitly accounting for the weaker horizontal.

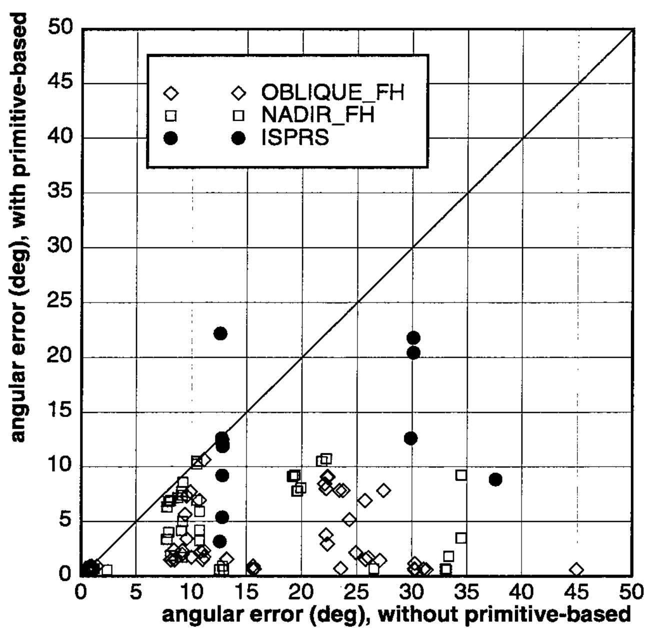


Figure 3-19: Effects of activating primitive-based detection

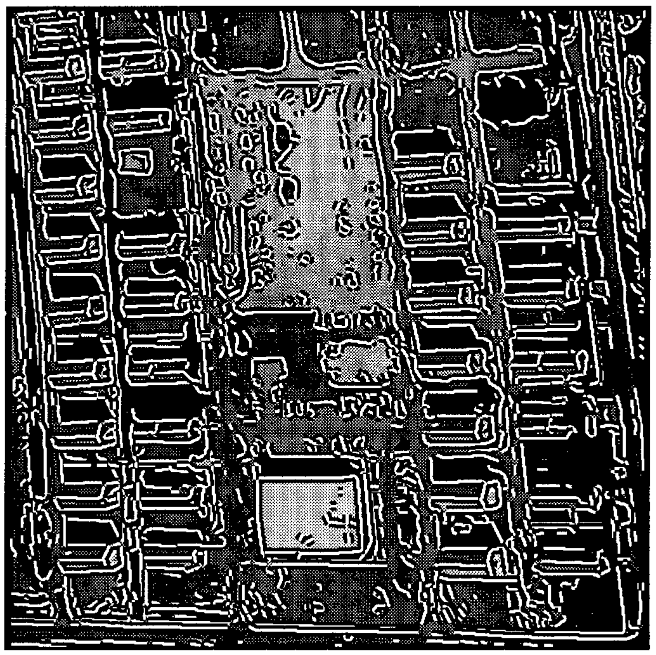


Figure 3-20: Example Fort Hood image

3.6.2.3. Probability masking and edge error modeling

The next set of experiments considers the effects of probability masking and the use of edge error models. For the following analysis, statistical estimation remains disabled, and primitive-based vanishing point detection is activated, in keeping with previous experimental results. To facilitate analysis, the evaluation of average angular error is broken into two components; error for horizontal line segments in object space, and error for slanted, or "peak," line segments. Horizontals are considered first.

Figures 3-21 through 3-24 depict average angular error as in previous diagrams, considering horizontal line segments in object space exclusively. These figures are logarithmically scaled along each axis, to better illustrate the change in error for small error values. Each figure shows the effect of activating one of the two edge error models, with probability masking turned on or off in each figure. Before proceeding with the analysis, note that both ISPRS images are present in Figures 3-21 and 3-23; they overlap at almost exactly the same coordinates, and hence appear as a single dot.

First, we observe that probability masking produces little or no change in angular error for horizontal edges in the Fort Hood images; this is due to the fact that horizontal edges are quite strong in the Fort Hood data, and natural textures are relatively absent from these scenes. Comparing Figure 3-21 with 3-22 and 3-23 with 3-24, we find that probability masking improves the orientation estimates for one of the ISPRS images, but the other image requires the use of an edge error model to correct the orientation estimate, which is quite poor (32 degrees of angular error) due to texture edges.

With either edge error model, the average angular error for horizontal line segments is less than 2.1 degrees in all cases; in most cases, the error is less than one degree. This analysis shows that horizontal orientations can be reliably estimated, even in cases with large amounts of textural noise, if either edge error model is employed. However, horizontal edges are often the longest edges in manmade structures; after accounting for noise, it should be expected that orientation estimates for these features would be reliable. We now turn to an analysis of the estimated orientations of peak (non-horizontal and non-vertical) edges, characteristic of slanted and gabled rooftops, and typically much shorter in length than the horizontal building edges.

First, we consider using the swath model of Section 3.5.1, without probability masking, in Figure 3-25. Here, the swath model improves average angular error in all cases, often dramatically; here, errors of greater than 70 degrees are reduced to no more than 20 after the swath model is activated. The results also cluster in an intuitive way; oblique images show errors of less than 8 degrees, while nadir images show greater errors, between 11 and 26 degrees. Since peak edges in nadir views will often appear as horizontals, it is to be expected that peak orientation estimates will be less reliable than their oblique counterparts.

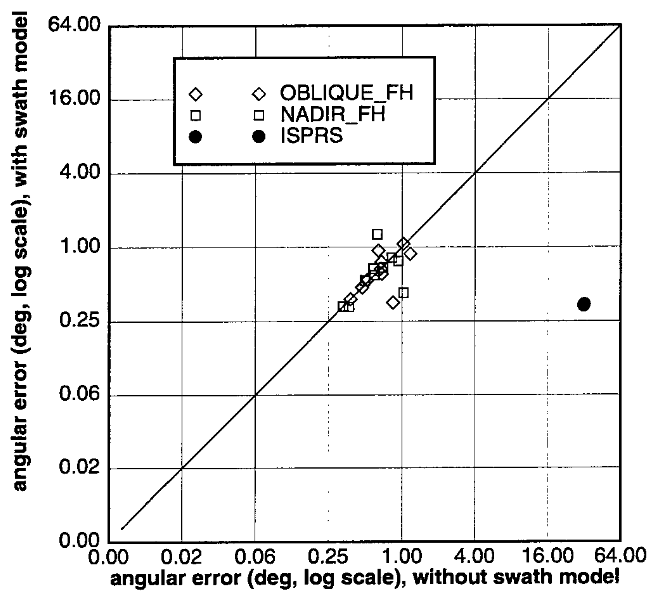


Figure 3-21: Effects of swath error model on horizontals, probability mask off

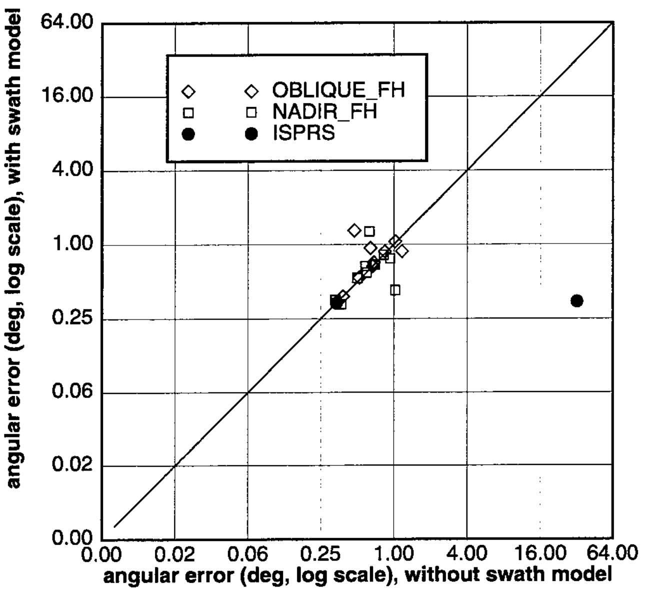


Figure 3-22: Effects of swath error model on horizontals, probability mask on

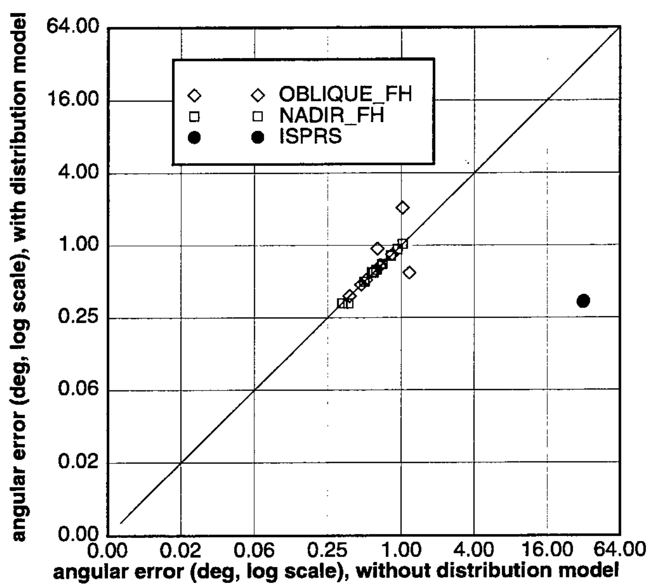


Figure 3-23: Effects of distribution error model on horizontals, probability mask off

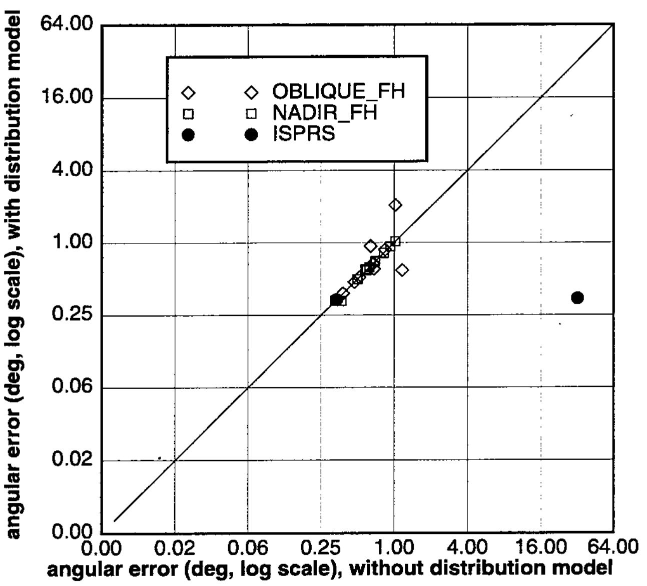


Figure 3-24: Effects of distribution error model on horizontals, probability mask on

Figure 3-26 shows the same experiment, this time with probability masking activated. Note that both ISPRS images appear as one dot in the graph, with nearly identical coordinates. The greatest x-coordinate in this graph is less than 40 degrees; contrasting this with Figure 3-25, we can see that probability masking without an edge error model makes a significant improvement, excepting the ISPRS images. However, activating the swath model in conjunction with probability masking gives mixed results; angular error in the ISPRS images is much worse. In these images, the true vanishing points for peak edges lie relatively close to the image bounds, where probability masking has a much stronger effect on the histogram. While probability masking has the advantageous effect of eliminating spurious histogram maxima within or near the projection of the image bounds on the sphere, it has the disadvantage of suppressing true maxima that might lie in or near the image bounds, as in these cases.

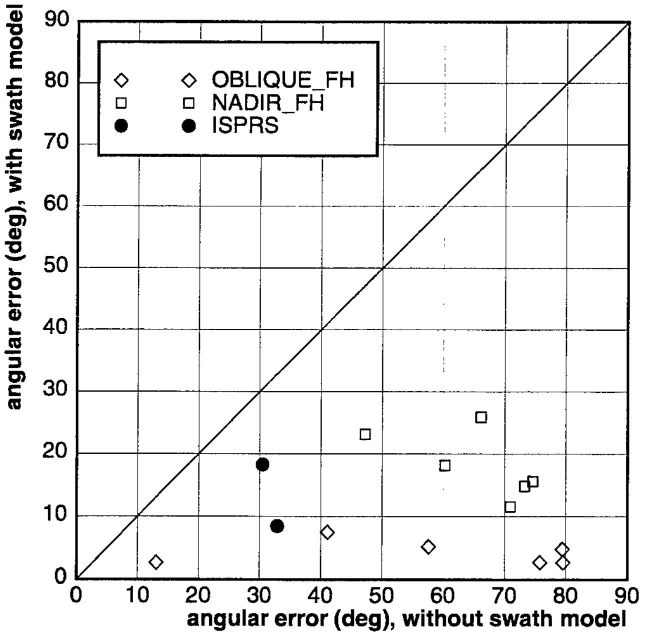


Figure 3-25: Effects of swath error model on peak edges, probability mask off

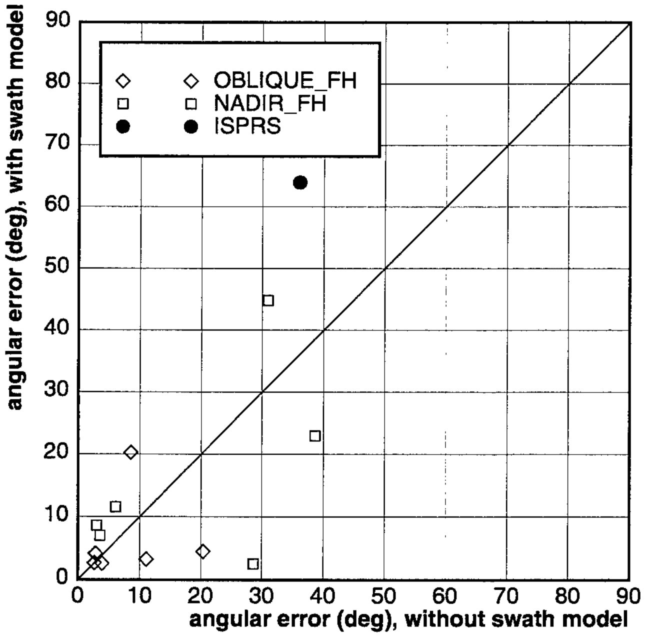


Figure 3-26: Effects of swath error model on peak edges, probability mask on

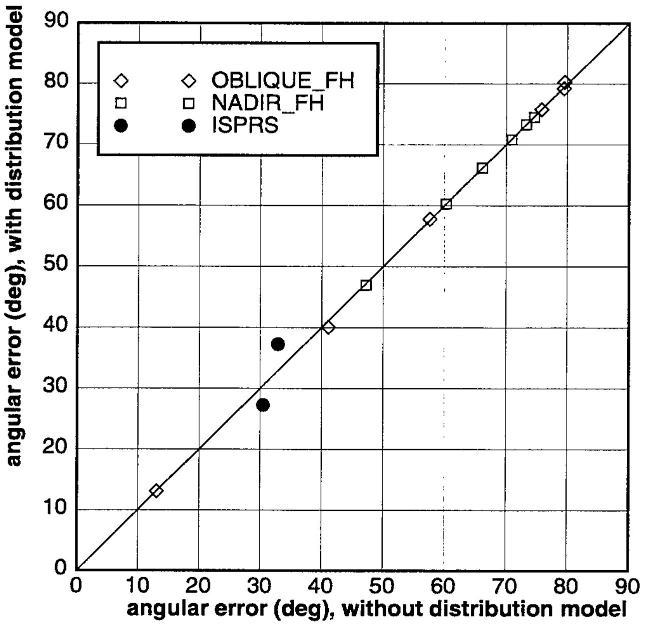


Figure 3-27: Effects of distribution error model on peak edges, probability mask off

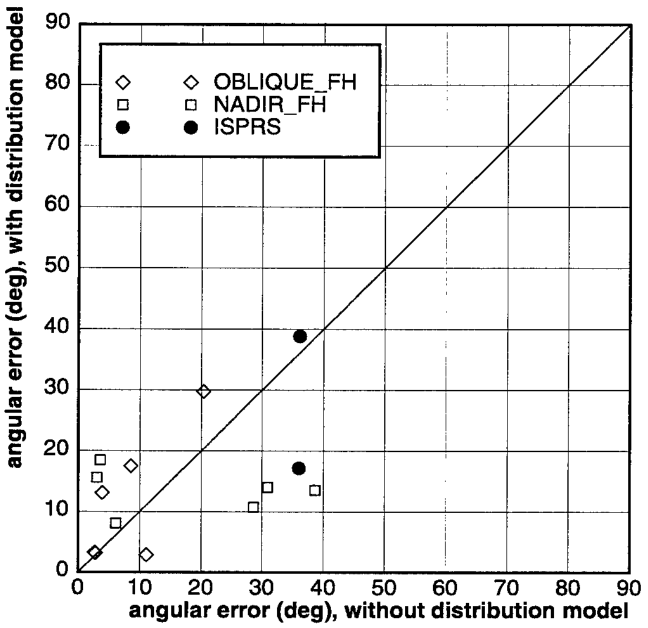


Figure 3-28: Effects of distribution error model on peak edges, probability mask on

Figure 3-27 considers the effect of the distribution model of Section 3.5.2, without probability masking. Unlike the swath model without probability masking, the distribution model has little or no effect on angular error in every case. This result can be explained by considering the effect of the distribution model on the Gaussian sphere histogram. The histogram can be converted to an image for viewing, where columns correspond to azimuth and rows correspond to elevation, and brighter pixels correspond to larger values in the histogram. Figure 3-29 shows the histogram for one test scene, where the distribution model has not been invoked, while 3-30 shows the histogram when the distribution model has been invoked, holding all other options constant. Although the distribution model has the effect of "smoothing" the histogram, the locations of the histogram maxima (the brightest pixels in the image) remain unchanged, and hence the detected vanishing points and average angular error remain unchanged.

Figure 3-28 shows the same experiment as Figure 3-27, this time with probability masking activated. As before, we observe that probability masking without the edge error model makes a significant improvement; all images show no more than 40 degrees of error, while all but three have greater than 40 degrees of error in Figure 3-27. As before, however, we also observe that the combination of the edge error model with probability masking leads to mixed results. These difficulties lie with the fact that probability masking operates independently of the edge error models, and does not take into account uncertainty in edge orientation and position. So, while weak and inaccurate vanishing point solutions generated by short edges can be corrected by probabilistic weighting, it is also the case that accurate vanishing point solutions generated by strong edges can be shifted away from their true values by the same weightings.

This set of experiments suggests two research issues worthy of future consideration. First, the comparison of Figure 3-25 with Figure 3-27 implies that a uniform distribution provides a better model of edge position and orientation error than the normal distribution, for the vanishing point detection task. Whether this is universally true or only true for the class of imagery addressed here is unclear. Second, Figures 3-26 and 3-28 show that the treatments of finite image extent and edge uncertainty are dependent on one another; an interesting question is whether these factors can be addressed in a single model, leading to consistently improved performance.

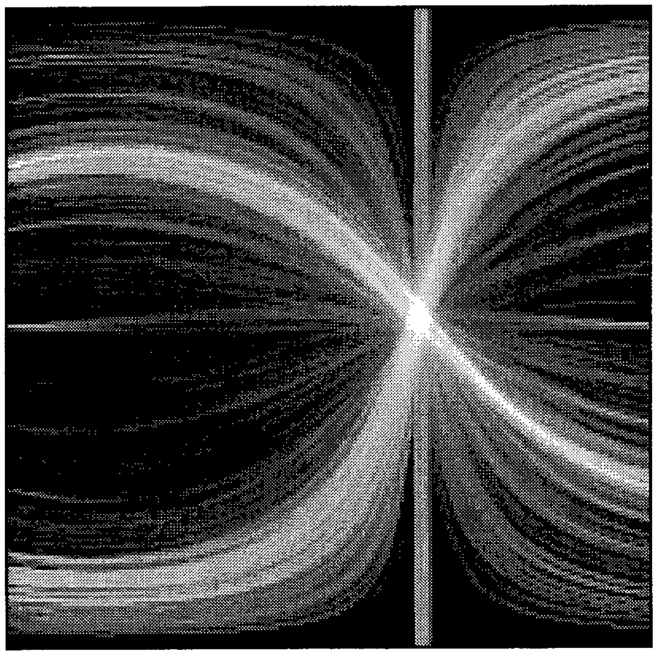


Figure 3-29: Histogram without distribution model

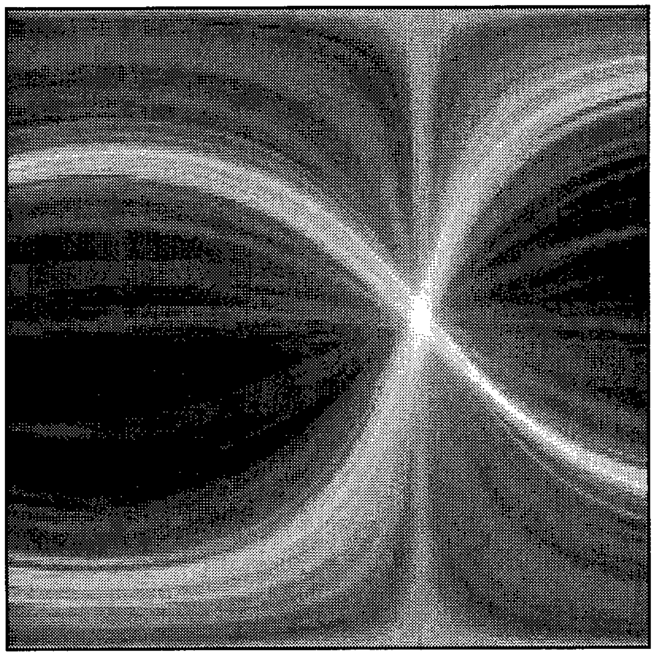


Figure 3-30: Histogram with distribution model

Another performance evaluation issue which remains to be addressed is the degree to which a vanishing point detector estimates the correct number of vanishing points for an image. To understand the issue, consider a vanishing point detection algorithm which ignores the image entirely, instead simply distributing points uniformly over the sphere, with a very small spacing ε between points. It follows that every orientation in object space is approximated by a vanishing point hypothesis with an angular error no greater than ε. However, an object detection algorithm will now have to process all of these hypothesized vanishing points as potential feature orientations, slowing down the algorithm significantly. Ideally, an automated vanishing point detection algorithm would not only generate accurate orientation estimates, but also generate only as many estimates as there are orientations of interest in the scene, and no more. In practice, PIVOT hypothesizes roughly two to three times as many vanishing points as the number that actually correspond to building structure. This suggests an interesting tradeoff between computation time spent during vanishing point detection, and time spent during object hypothesis generation; this tradeoff is a topic for future exploration.

3.7. A summary of vanishing point analysis

In this chapter, the rectangular and triangular volume primitives were introduced as candidates for modeling building structure in aerial mapping photography. Vanishing point analysis was then motivated as the means by which 3D orientation information about these primitives could be obtained from monocular imagery. Two new techniques were introduced for enhancing the traditional Gaussian sphere histogramming technique for vanishing point detection; primitive-based detection and interpretation plane (edge) error modeling, the latter of which encompassed two distinct methods for modeling the uncertainty of edge position and orientation. In the previous section, a comprehensive analysis of angular error was presented for these techniques and two other techniques, from which the following conclusions can be drawn:

- Statistical estimation of interpretation plane normals [20], which attempts to refine vanishing point hypotheses, fails in the absence of strong perspective effects.
- Primitive-based vanishing point detection, which uses geometric knowledge about objects of interest, consistently improves vanishing point accuracy.
- Probability masking [61], which attempts to account for bias induced by finite image extent, is effective in reducing angular error when edge uncertainty is not modeled independently.
- The swath error model of Section 3.5.1, without the use of probability masking, provides robust results on the 22 test images used in this work, achieving angular errors of less than 1.3 degrees for horizontal edges, and angular errors of less than 8 degrees for short peak edges in oblique views. Previous methods have not addressed non-horizontal edges.

By using the swath error model and primitive-based vanishing point detection, it is possible to compute vanishing point information suitable for object detection and delineation. Figure 3-31 shows another test image of Fort Hood, taken from an oblique angle. Figure 3-32 shows the results of edge detection [78] on this image.

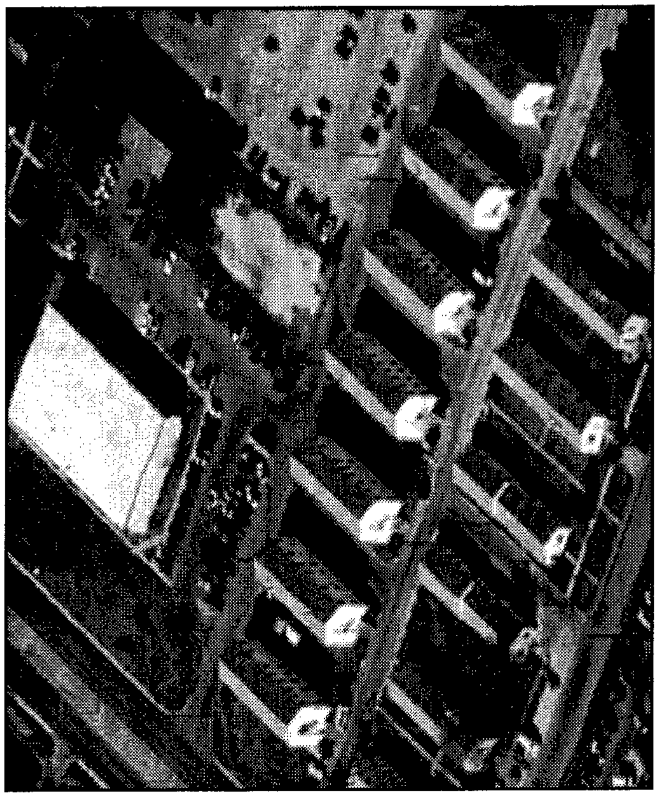


Figure 3-31: Fort Hood oblique image

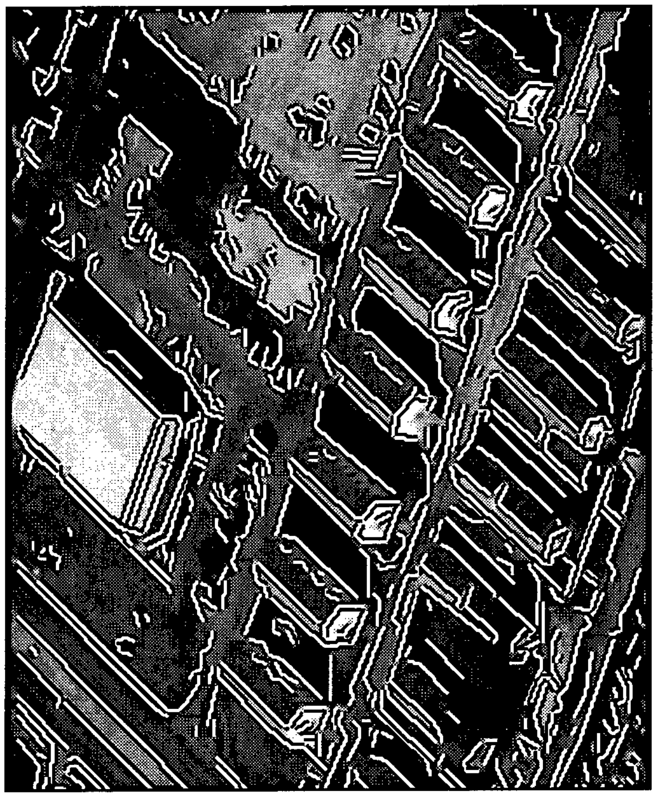


Figure 3-32: Edge detection results

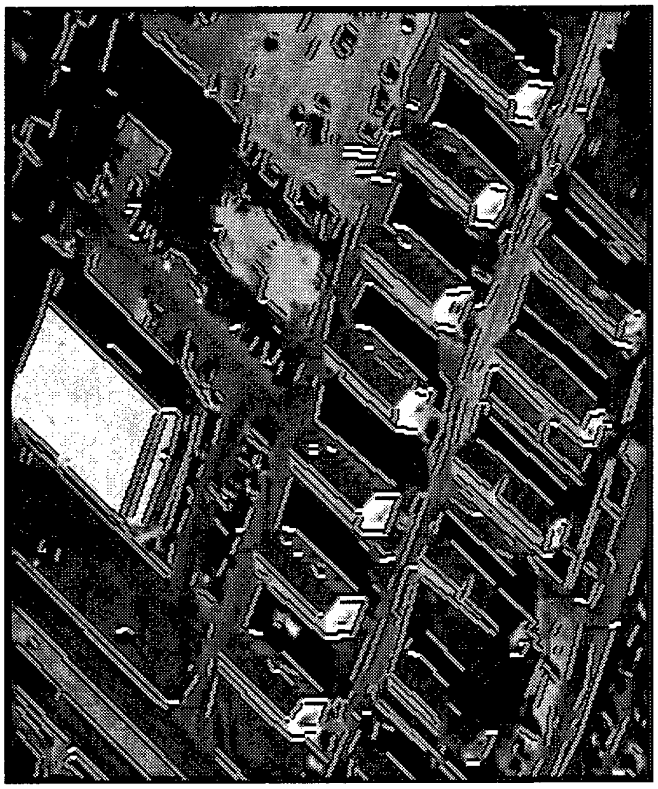


Figure 3-33: Horizontal and vertical segments

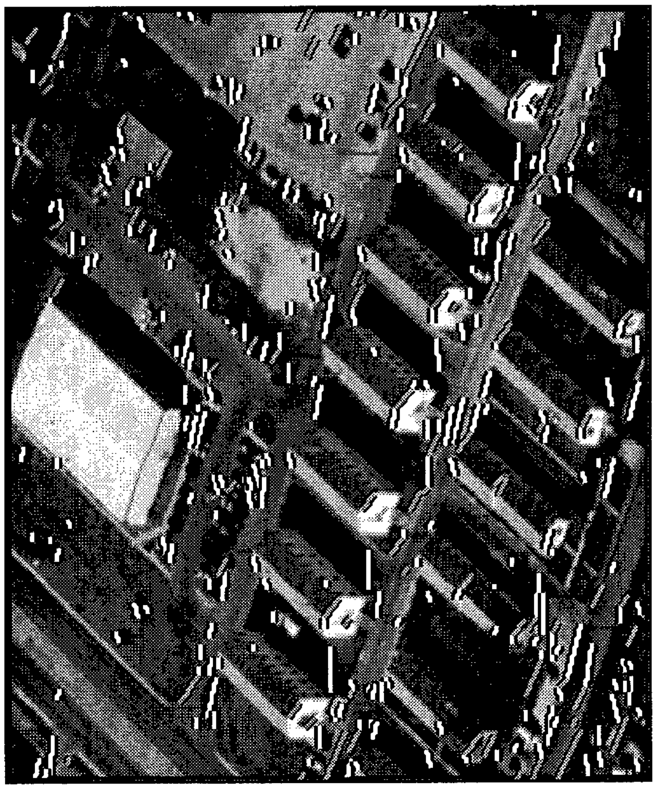


Figure 3-34: Slanted roof segments

After automatically computing vanishing points using the swath model and primitive-based detection, each edge is tested to determine whether it lies on a line through a vanishing point. Figure 3-33 shows the edges labeled as horizontals and verticals in object space (**v**, **h1**, and **h2** in Figures 3-1 and 3-2); Figure 3-34 shows the edges labeled as slanted peak roof edges in object space (**p1** and **p2** in Figure 3-2). In particular, note that many of the building edges are properly labeled as horizontal, vertical, or slanted. This detailed orientation information is well-suited to existing object extraction techniques [67], and illustrates the utility of Principle 1.

One difficulty with any Gaussian sphere histogramming approach for vanishing point detection is its assumption of a "grid-world," where objects of interest lie on an orthogonal grid. For many urban environments, such an assumption is reasonable; for suburban areas, where houses may encircle a cul-de-sac, the approach will break down, since any horizontal orientation may be represented by only one house, and natural textures in the scene will likely produce greater maxima in the histogram. Generalizing the method to handle scenes with a large range of object space orientations is a topic for future work.

One approach to this problem is to analyze the strength of the vanishing point solutions relative to the noise level of the histogram. If the vanishing points are not statistically significant, then the image could be subdivided and the vanishing point algorithm could be employed on each of the image fragments. Since fewer orientations would be represented in each fragment, specific object orientations could be extracted from each fragment. Open questions are whether such a subdivision could be performed such that objects would be uniquely segregated among fragments; whether such a method would be effective in the presence of occlusions, where segregation would be impossible; and whether the method could be expected to cope with increasingly weak perspective effects as the size of the subdivisions decreases.

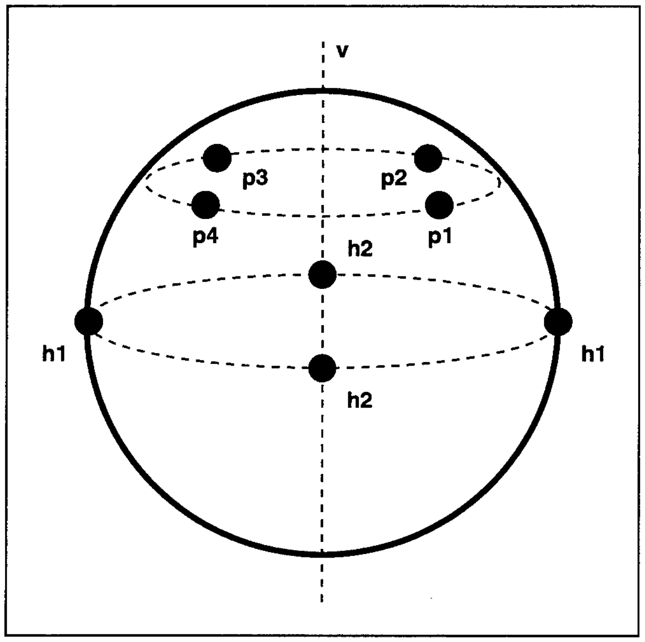


Figure 3-35: Orientations of a triangular gable

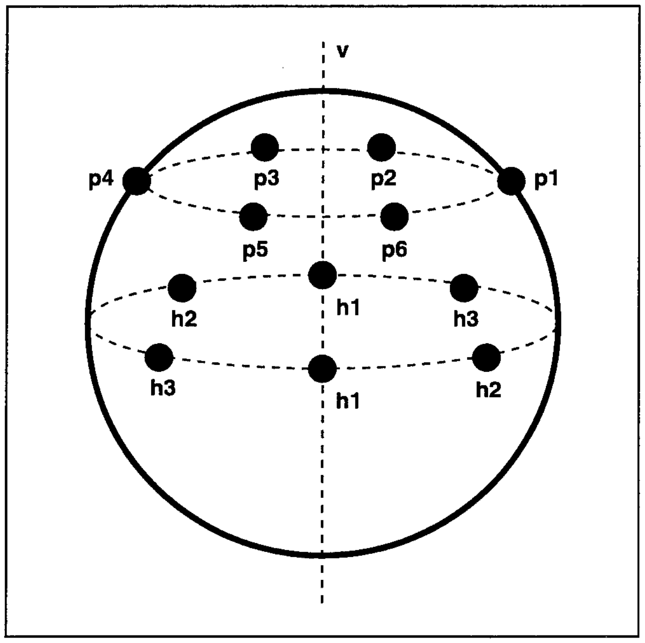


Figure 3-36: Orientations of a hexagonal pyramid

Another question concerns the sensitivity of the vanishing point detection methods as object shapes and symmetries become increasingly complex. Figures 3-35 and 3-36 depict the vanishing point patterns for a gabled triangular prism, where the triangular facets slope inward, and for a hexagonal pyramid. The vanishing point structures for these primitives are more complex than those for PIVOT's rectangular volumes and triangular prisms. Since the experiments in Section 3.6 showed that slanted peak orientations were less accurate than horizontals, it is natural to ask whether the gabled slanted lines or slanted pyramidal lines would be even less accurate.

One aspect of the vanishing point techniques which has not been addressed in this work is the choice of quantization resolution. Throughout this chapter, the Gaussian sphere was quantized at a resolution of one degree, leading to a 180×180 histogram. This value was chosen to give accurate vanishing point orientation estimates and the ability to distinguish vanishing points for nearby orientations, while preventing spreading effects in the histogram. Spreading is caused by line segments with very similar world orientations mapping to separate nearby buckets in the histogram instead of the same bucket, thereby weakening maxima in the histogram. Whether the methods described in this chapter are sensitive to resolution choice is an open question; automatic selection of an appropriate resolution is also an interesting topic for further research. A first step towards addressing these questions might begin with a formal error model for Gaussian sphere quantization, accounting for deviation in an edge.

Another topic for future work is a generalization of the primitive-based detection technique of Section 3.4. If the relative edge orientations of an object of interest were fixed, then instead of looking for only orthogonal horizontals or symmetric peaks, the Gaussian sphere could instead be scanned for the complete set of vanishing points for the object, allowing the use of arbitrary polyhedra for primitive models. This can be viewed as a mapping from the original Gaussian sphere to a new sphere, where each bucket on the new sphere represents an orientation for the object, and the value of the bucket is the sum of the appropriate buckets on the original sphere. A maxima on the new sphere then represents the best solution for all vanishing points associated with the object. This approach may lead to more accurate solutions for objects whose shape is fully specified, and alleviate the problems associated with complex vanishing point structures such as those depicted in Figures 3-35 and 3-36.

Two related research questions concern the applicability of vanishing point techniques: what are the performance limits for the vanishing point detection algorithm, and what classes of sensors can be handled by the vanishing point methods described here? The former question hinges on several related factors, including effective image resolution, image content, edge quality, and histogram resolution, among others. A theoretical model of these factors would be useful in determining the likelihood of accurate performance for any given scene. The latter question is geometric in nature; imaging devices such as pushbroom scanners, which obey central projection on a line-by-line basis, require different geometric models. The development of vanishing point methods for alternate sensor platforms is an interesting area for future work.

The vanishing point detection algorithms and analysis presented in this chapter extend the state-of-the-art to imagery with weak perspective effects and noisy edge data. As the need for semi-automated and automated object recognition systems increases, particularly in cartographic domains, photogrammetrically-derived cues for object extraction become increasingly important as a source of intrinsic geometric constraints [71]. Vanishing points represent one such cue, and the ability to reliably detect and exploit them is an important step towards robust and efficient object extraction from real imagery. In the next chapter, we consider feature extraction algorithms which make use of vanishing points to construct object hypotheses.

Chapter 4
Geometric Constraints for Hypothesis Generation

As Chapter 3 illustrated, vanishing points serve as useful tools for qualitative and quantitative descriptions of edge geometry. Quantitatively, they provide estimates of line orientation in object space; qualitatively, they can be employed to constrain the search for primitives in hypothesis space. The techniques developed in Chapter 3 exploited the primitive geometry to hypothesize plausible 3D orientations, or labels, for each line segment in the image.

The next step in a data-driven approach to object detection is to create intermediate features which provide the necessary elements for primitive generation. In essence, this amounts to a hierarchical search for primitive instances in the image data. At each step of the hierarchy, increasingly complex features are constructed from the simpler features, until it is possible to directly hypothesize primitive volumes.

A key issue in feature generation is the combinatorics of search in hypothesis space. It is well understood that brute force attempts at feature generation, where all possible combinations of edges are evaluated, leads to an exponential growth in the size of the search space. Geometric constraints play a central role in feature generation, by limiting this growth to a manageable level. In this light, it should not be surprising that Principles 3 and 4, together paraphrased as *"use geometric constraints as soon as possible, but no sooner,"* play a significant part in hypothesis generation.

The main thrust of this chapter is the development of intermediate features which are amenable to constraints based on vanishing point and vanishing line geometry. Two intermediate features are considered in detail: *corners* and *2-corners*. These features are produced by combinations of the labeled line segments, and subjected to a variety of geometric constraints which have the dual purposes of reducing the combinatorics of search and ensuring that these features can lead to legal primitive instances. From these features, primitives are instantiated to produce an initial set of object hypotheses. Throughout this process, geometric constraints based on vanishing point information are used extensively to lead to efficient, and more importantly, robust generation of primitive hypotheses.

4.1. Corner detection

Corners are a common and widely-used intermediate feature in vision, especially in scenes where polyhedral objects are dominant. This is due to several advantageous properties of corners. They can easily be detected, either by convolution of image intensities with a corner detecting template or by range search on edge endpoints; they are the simplest features that can be formed by line segments; and, together with their component edges, they constrain the search for related corners which might comprise a facet of an object.

In the aerial image domain, much research has relied on corners to guide the search for building hypotheses. A thorough discussion of corner analysis for aerial images was presented by Huertas and Nevatia [42]. Their system linked line segments which were nearly orthogonal in the image, if the nearest endpoints of the line segments were within 5 pixels of one another. In addition, they used a set of local heuristics to determine whether the arms of T-intersections should be broken into two pieces by the stem of the intersection. They also describe the difficulties in reliably handling T-intersections, due to the number of possible interpretations which cannot be disambiguated from a single view.

Nearly all of the work since 1988 to the present has utilized corners as an intermediate feature (Tables 1-2 and 1-3 in Chapter 1 give brief descriptions of that work). In particular, many of these systems have utilized corners in nadir views of urban scenes, since right-angled corners of rooftops in the scene project to nearly right-angled corners in the image, which can be easily detected. If buildings are assumed to be four-sided, then rooftop generation involves a search for right-angled corners which can form a closed rectangle.

The difficulty with these assumptions is that they fail in oblique views of a scene, or when objects are not exclusively rectangular volumes. From arbitrary viewpoints, right-angled corners in the scene do not, in general, appear as right-angled corners in the image, and objects may not have right-angled corners at all. These assumptions violate the principles of object detection set out in Chapter 2, because they make restrictions on the image and scene. Instead, a more general approach is employed in this work, using the vanishing point information associated with each line segment in the image.

The first step in this approach is to use a 2D range search on line segment endpoints to generate corner hypotheses. A bridged range tree [59, 87, 106] is the data structure used for the search, as it achieves the optimal query time of $O(\log n)$, with $O(n \log n)$ preprocessing time and space consumption of $O(n \log n)$. Every endpoint of every line segment is entered into the bridged range tree, along with a line segment identifier. Then, for each endpoint q, a query rectangle is constructed with q at the center and all points inside the rectangle are returned by the range query. The identifier for each of these points is then used to access its line segment, which is linked to q's line segment to form a corner.

A fixed-size query rectangle is sometimes used for corner detection in the range search paradigm, but the selection of an appropriate size is difficult, since there can be noise or occlusions at or near corner points, which forces the use of a larger query rectangle and increases the number of corners generated by an endpoint.

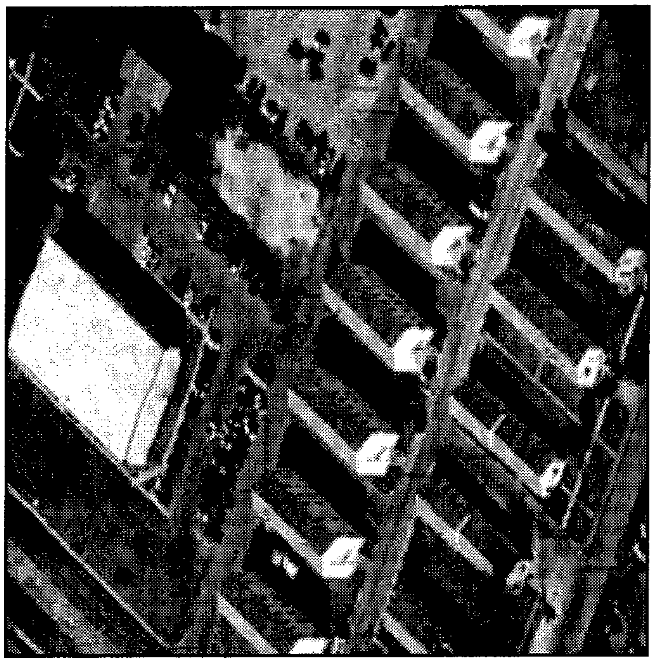


Figure 4-1: Fort Hood example image

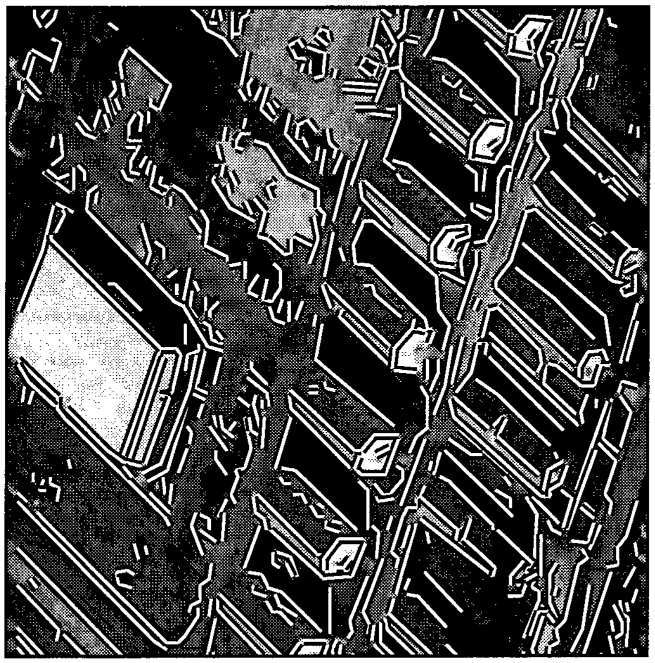


Figure 4-2: Results of line extraction

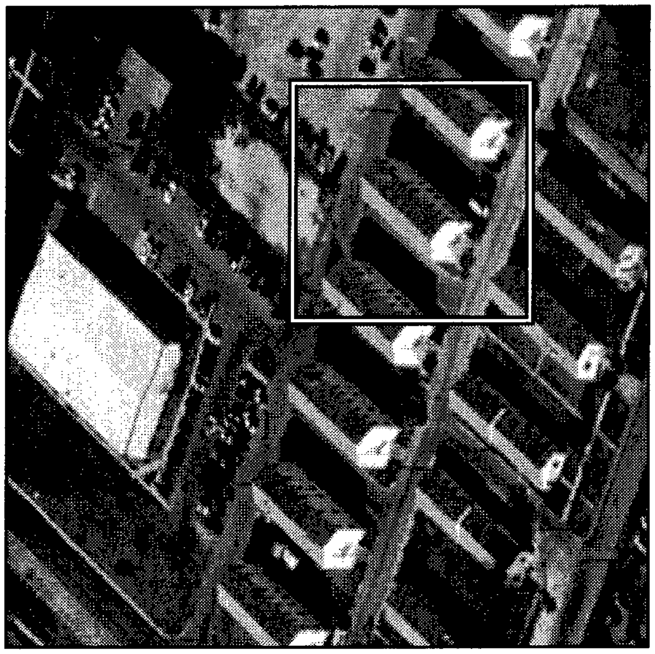


Figure 4-3: A subarea of the image

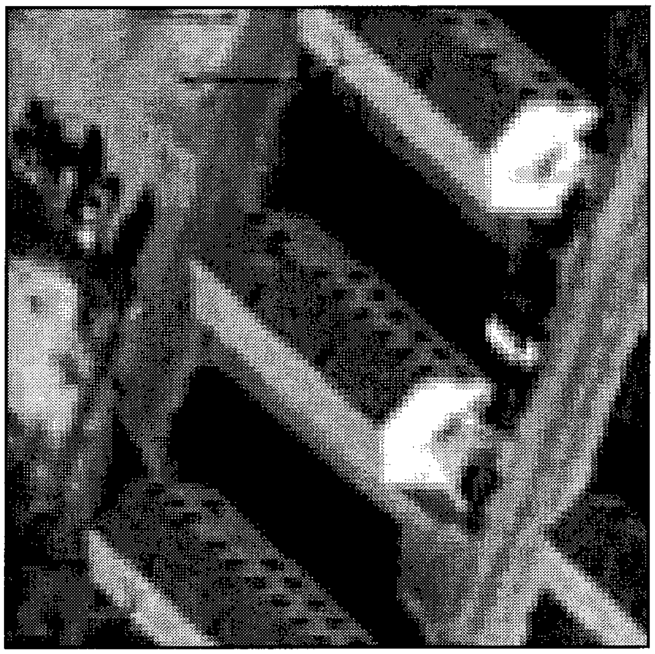


Figure 4-4: A close-up of the subarea

Instead, a variable-size query box is used for each endpoint q, with radius equal to the length of q's line or a maximum value (100 pixels in PIVOT), whichever is smaller. This allows areas proportional to line segment length to be searched, while the maximum value places an upper bound on the size of the area to be searched. This ensures that corner detection on long segments does not require searching large regions of the image which are unlikely to contain potential corner-forming segments.

However, this basic approach can still generate a huge number of corners from accidental alignments of line segments, and the combinatorics become even worse when vanishing point interpretations of these corners are considered. Recall from Chapter 3 that the vanishing point detection method generates a set of vanishing points in the image plane. Each line segment extracted from the image can be tested to see if it lies on a vanishing line through any of the vanishing points, and labeled with the object space orientation associated with the vanishing point through which its extended line passes. It is possible for each line segment to be assigned multiple vanishing point labels, which means that each segment has multiple orientation interpretations which must be considered in later processing.

For the purposes of this chapter, an *interpretation of a line segment* will be an assignment of one of the possible vanishing line orientations to the line segment. An *interpretation of a corner* comprised of line segments *a* and *b*, with *m* and *n* interpretations respectively, will be an assignment of one of the *mn* combinations of orientation labelings to the corner. Generalizing to intermediate features comprised of multiple line segments, we say that an *interpretation of a feature* is an assignment of possible orientation labels to each of its component line segments. It should be clear that if a line segment has *m* interpretations in the average case, then the addition of one line segment to another to form a corner, or to a corner to form a more complex intermediate feature, increases the total number of interpretations of that feature by a factor of *m*, and the construction of increasingly complex intermediate features from line segments becomes exponential in the number of interpretations.

Consider Figure 4-1, a portion of an image of Fort Hood, Texas, distributed under the RADIUS research program. For this image, the vanishing point detection process found 13 vanishing points. One was the vertical vanishing point, computed automatically from the camera model. Four were made up of two pairs of orthogonal horizontal vanishing points; one pair of the two did not correspond to building structure in the scene. The eight remaining vanishing points consisted of four pairs of slanted "peak" vanishing points, one pair for each horizontal. After edge and line extraction and vanishing point analysis, 1356 line segments were produced, with 3257 interpretations (on average, 2.4 interpretations per line segment). Figure 4-2 shows the results of line extraction for this image.

Figure 4-3 bounds a small section of the image, which is shown at higher resolution in Figure 4-4; this section will be used as a running example to illustrate the effects of various geometric constraints on corner generation. Figure 4-5 shows the lines extracted for this section of the image. One line segment in the center of this figure, corresponding to the peak ridge of a rooftop, has been shaded. Successive figures will show the corners which use this line segment as one arm of a corner.

Figure 4-6 shows the initial set of corners generated from the shaded edge in Figure 4-5. There are 90 corners in this image, with 1352 interpretations (averaging 15 interpretations per corner). For the larger image, Figure 4-1, 8097 corners were generated with 129385 interpretations (averaging 16 interpretations per corner). As this example clearly shows, a large number of corners can be hypothesized in the range search phase, most of which have no basis in the scene, yet which contribute large numbers of interpretations. Section 4.2 considers the use of vanishing point information and relative distance constraints to prune corners which do not correspond to any legal corner in a primitive, and to prune corners which are created from spatially unrelated line segments.

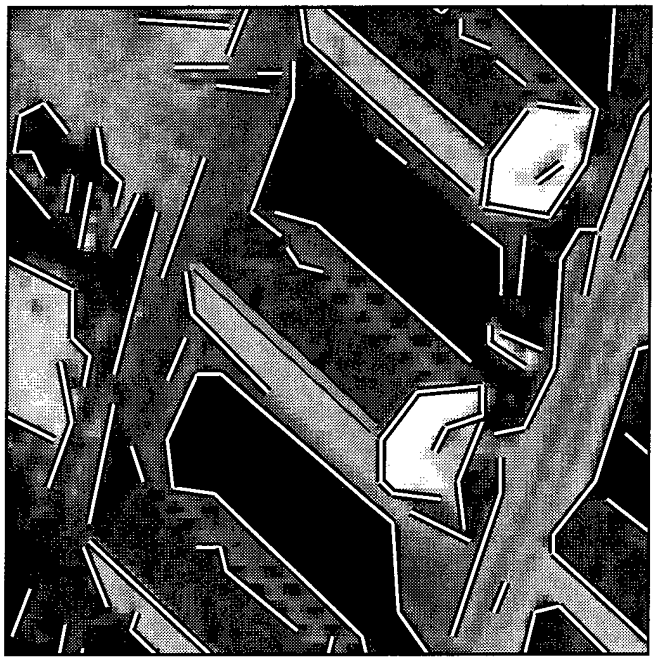


Figure 4-5: Line extraction on the subarea

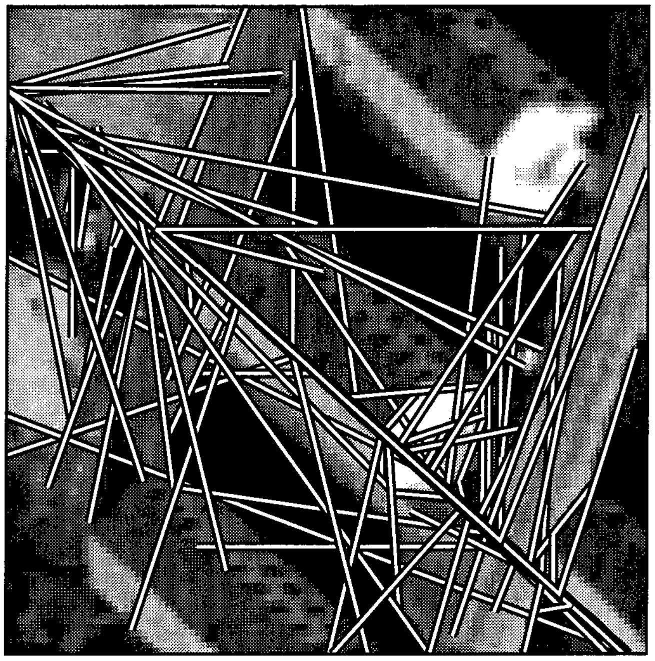


Figure 4-6: Initial corner generation results

4.2. Corner constraints

The key to reducing the number of corner interpretations which will be accepted for later analysis is the observation that only certain combinations of line orientations lead to legal corners in the primitives. A trivial example is the pairing of two vertical lines to form a corner; this is clearly illegal, since no corner in either primitive (or any primitive, for that matter) can be formed by two vertical lines. Such corners can be discarded immediately. A rudimentary version of this idea was employed in earlier work [67].

The geometric constraints are implemented as tests for each potential corner pair to see if the pair is a legal corner for either of the two primitives. If it is not, it is rejected. Since each edge can pass through multiple vanishing points, and hence have multiple possible orientations, each orientation must be tried. This implies that two line segments can generate zero, one, or many corners, depending on the number of legal orientation pairings. The legal pairings for the two primitives in Section 3.2 are as follows:

- A vertical (**v**) line segment and a horizontal (**h**) always produce a legal corner, since these can occur in a rectangular primitive.
- Since horizontals are found in pairs, where the two horizontals **h1** and **h2** are orthogonal, **h1** and **h2** always produce a legal corner which can occur in either primitive.
- Since any pair of slant vanishing points **p1** and **p2** are found with respect to a horizontal pair **h1** and **h2**, the corner combinations **p1–h1**, **p1–h2**, **p2–h1**, and **p2–h2** form legal corners which can occur in a triangular primitive.
- Any other combination of orientations is illegal, since these combinations do not occur in either primitive.

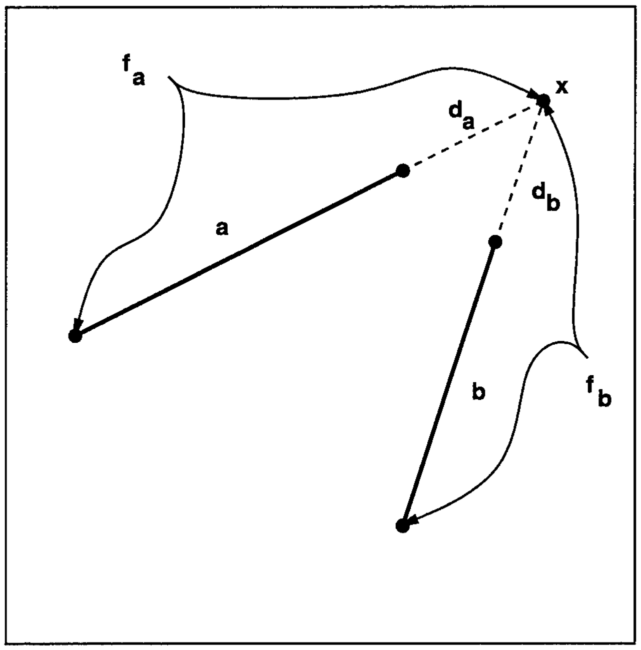


Figure 4-7: Regular corner intersection

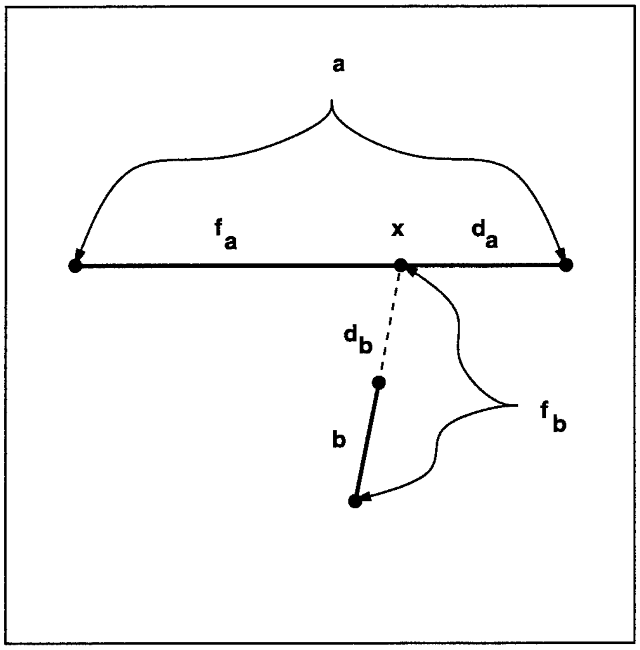


Figure 4-8: T-intersection

Given a list of n vanishing points produced by the primitive-guided Gaussian sphere analysis in Chapter 3, it is trivial to construct an $n \times n$ array A of booleans, where the value of $A[i][j]$ indicates the legality of pairing vanishing point i with vanishing point j. Testing a corner then reduces to a single array access.

For the 90 corners in Figure 4-6, the application of vanishing point constraints reduced the number of interpretations from 1352 to 406. Although no corners were discarded in this section of the image, since there was at least one valid interpretation of each corner, a few were discarded for the entire image. The number of corners in the entire image dropped from 8097 to 8032, but more importantly, the total number of interpretations fell from 129385 to 35736, lowering the average number of interpretations per corner from 16 to 4.4.

Many corners can be eliminated by using purely geometric constraints, which is in keeping with the principles described in Chapter 2. However, many spurious corners remain, since there can always be alignments of unrelated edges which happen to possess legal vanishing point geometries. For further reduction in the number of corners, distance constraints can be employed which check to see if the gap between two edges is sufficiently small to allow a corner to be produced. Rather than use a fixed threshold for the allowable gap, which can fail due to the same difficulties encountered during range search, relative distance constraints are employed. There are several versions of these constraints which can be used; each one is discussed in turn.

Before discussing the relative distance constraints, it is useful to refer to Figures 4-7 and 4-8, which depict two typical situations. In Figure 4-7, two line segments extend to intersect at a point which does not lie on either segment. In Figure 4-8, two line segments intersect at a point which lies on one of the line segments; this case is referred to as a *T-intersection*. In both diagrams, a and b denote the lengths of each line segment; d_a and d_b denote the distance from the intersection point x to the closest endpoint of each line segment; and f_a and f_b refer to the distance from the intersection point to the furthest endpoint of each line segment.

The *weak* corner distance constraint simply compares the length of each corner edge with its distance to the computed intersection point. If the length of either edge is less than its distance to the intersection point, then the corner is rejected. Stated more precisely:

$$\text{if } (d_a > a \text{ or } d_b > b) \text{ reject} \tag{4.1}$$

The *strict* corner distance constraint is a stronger version of the weak constraint. In addition to comparing the length of each line segment with its distance to the intersection point, it also compares the length of each segment with the distance of the *other* segment to the intersection point. Essentially, neither line segment can be shorter than the shortest distance of either segment to the intersection point. More precisely:

$$\text{if } (\max(d_a, d_b) > a \text{ or} \max(d_a, d_b) > b) \text{ reject} \tag{4.2}$$

The *strict-T* corner distance constraint is a slightly weakened version of the strict constraint, which allows T-intersections to pass. The strict constraint would reject the corner depicted in Figure 4-8, since $d_a > b$. A solution to this problem is to perform the strict test only when the intersection point does not lie on either line segment, and default to a weak test when a T-intersection is a possibility. More precisely:

$$\text{if } (d_a > a \text{ or } d_b > b \text{ or} (d_a > b \text{ and } f_a > a) \text{ or} (d_b > a \text{ and } f_b > b)) \text{ reject} \tag{4.3}$$

Figure 4-9 shows the result of applying the weak corner distance constraint to the running corner example. This constraint reduced the number of corners from 90 to 21, and the number of interpretations from 406 to 105. While the average number of interpretations per corner remains the same, the number of corners drops significantly. For the entire image (Figure 4-2), the number of corners was reduced from 8032 to 2256, and the number of interpretations dropped from 35736 to 10331. This illustrates the power of even simple distance constraints for pruning corners. However, many spurious corners still remain.

Figure 4-10 shows the results of applying both the strict and strict-T distance constraints on the running example; in both cases, the number of corners was reduced from 21 to 7, and the number of interpretations was reduced from 105 to 34. On the larger image, the application of the strict constraint reduced corners from 2256 to 1313, and interpretations from 10331 to 5943. The strict-T constraint did not produce as steep a reduction, which was to be expected since the constraint allows T-intersections to pass; corners were reduced from 2256 to 1356, and interpretations fell from 10331 to 6158. In either case, however, the improvement over the weak constraint is significant. In particular, note that corners corresponding to both facets of the peak roof still remain in Figure 4-10, after the vanishing point and relative distance constraints have been applied. This illustrates the power of Principle 3; by applying geometric constraints as early and often as possible, nearly all spurious corners were discarded, while corners corresponding to actual scene structure were allowed to pass.

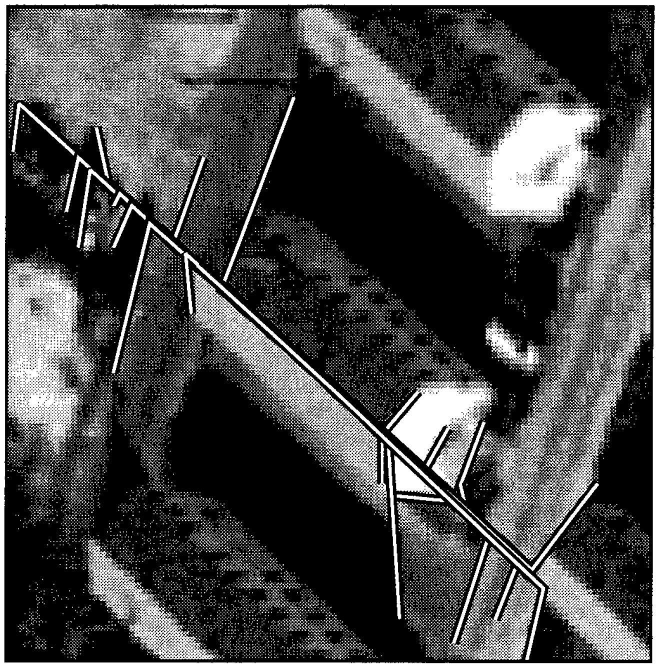


Figure 4-9: After weak corner distance constraint

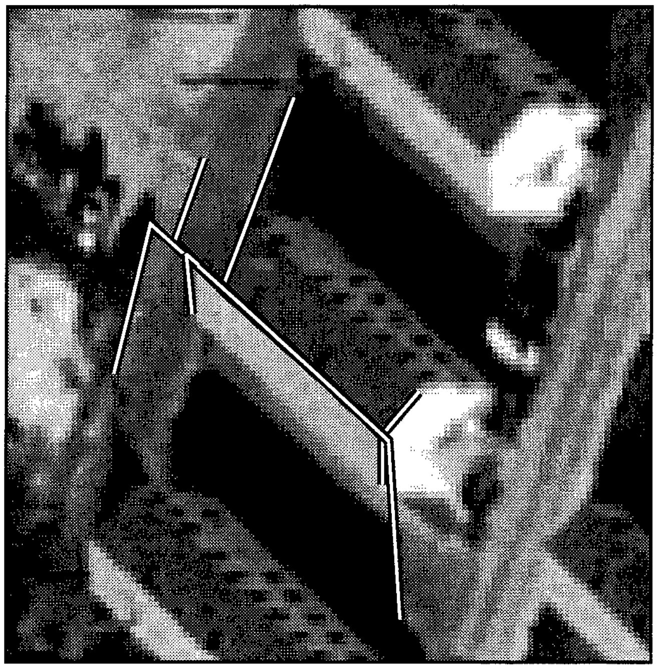


Figure 4-10: After strict corner distance constraints

It should be noted in passing that Principle 4 has been violated to some degree by the relative distance constraints; it is still possible that two arms of an object corner could be separated by a gap larger than the length of either arm. Shadow occlusions, natural vegetation cover, and texture patterns could all cause large gaps during edge and line extraction, resulting in the rejection of a valid corner corresponding to actual object structure. As mentioned in Section 2-11, strict adherence to Principle 4 is difficult to achieve without incurring combinatorial difficulties. Nevertheless, the relative distance constraints presented here represent an improvement over absolute distance constraints which use fixed window sizes, both in their flexible ability to find corners in the presence of noise, and in the ability to operate without user-specified parameters.

Before continuing to the next intermediate feature in the feature hierarchy, it is worthwhile to revisit the issues involved in handling T-intersections, illustrated earlier in this chapter by Figure 4-8, and in Chapter 2 by Figures 2-6 and 2-7. As Huertas and Nevatia [42] indicated, the difficulty in monocular analysis of T-intersections arises in determining whether the arms of the T should be treated as a single line segment corresponding to an unbroken line in object space, or whether the arms should split at the stem of the T, corresponding to a trihedral junction in object space. Their solution used heuristic tests with fixed pixel distances to make this decision; in particular, referring to Figure 4-8, they broke the arms at point x if $b>5, f_a>5$, $d_a>5$, $d_b<5$, b and $f_a d_a$ were nearly orthogonal, and if b did not already form a corner with another line segment.

The solution used in PIVOT is to always generate both corners at a T-intersection, breaking the arms of the T-intersection at point x to produce the corners $f_a b$ and $d_a b$ (again, see Figure 4-8). However, there is one restriction: d_a, the shorter arm, is only allowed to form a corner with b and no other segment. While this also violates Principle 4, it is necessary to prevent serious combinatorial difficulties, in cases where several stems intersect one long line segment. In practice, this approach is implemented by first finding all possible T-intersections with the range search scheme described earlier, and adding the short arm line segments (d_a in the figure) to the original collection of line segments. Corner generation then proceeds as before, with the additional constraint that short arm line segments form corners only with the stems that created them. In this way, both corners at a T-intersection will be generated.

Handling T-intersections increases the number of line segments, corners, and interpretations. The number of line segments for Figure 4-2 increases from 1356 to 1958 when short arm edges are added, an increase of 602 segments, with a total of 4784 interpretations (2.4 interpretations per segment). The number of corners increased from 1356 to 1469, an increase of 113 corners, with an increase in interpretations from 6158 to 6674. Note that the number of new corners is much less than the number of new short arm line segments, indicating that most new T-intersection corners do not satisfy the vanishing point constraints. For the small image section, the results are unchanged (7 corners with 34 interpretations).

Figure 4-11 shows the final set of corners for the Fort Hood image, after the use of vanishing point constraints, the strict-T distance constraint, and T-intersection corner generation. Table 4-1 summarizes the incremental effects of the geometric constraints discussed in this section as they are added to PIVOT's corner generation module. The combination of the strict distance constraint and the vanishing point constraints provides the most significant pruning power, but it does not allow T-intersections to pass through. The strict-T distance constraint allows the longer corner of a T-intersection, and T-corner generation creates the shorter corner of a T-intersection, accompanied by slight increases in the number of corners and interpretations.

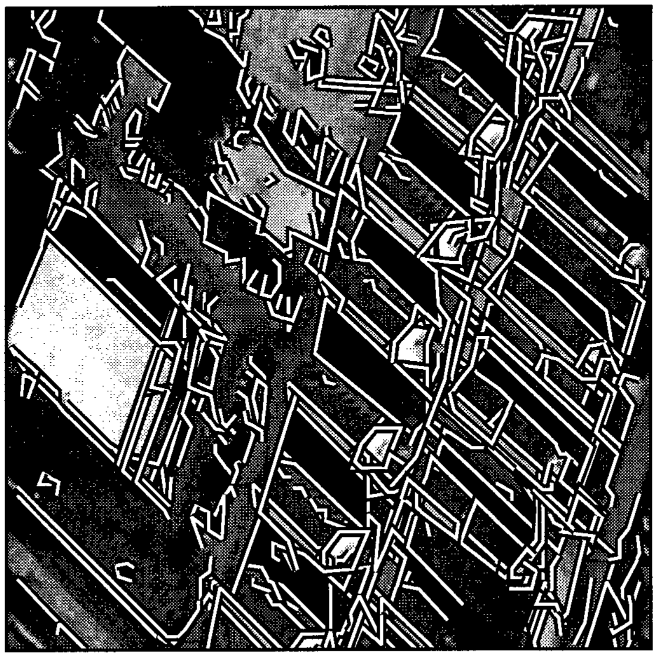


Figure 4-11: Final corner generation results

constraints	# of corners	# of interpretations
none	8097	129385
vanishing point constraints	8032	35736
vanishing point constraints + weak distance constraint	2256	10331
vanishing point constraints + strict distance constraint	1313	5943
vanishing point constraints + strict-T distance constraint	1356	6158
vanishing point constraints + strict-T distance constraint + T-corner generation	1469	6674

Table 4-1: Effects of geometric constraints on corners

The most important features to observe here are the impact of vanishing point constraints on the number of interpretations, and the impact of relative distance constraints on the number of corners. The combination of these two types of constraints allows corner generation to produce only those corners which, based on vanishing point geometry and local distance geometry, can possibly lead to instances of the primitive models. These constraints give significant savings in the number of corners and interpretations which must be considered by later processing phases.

4.3. 2-corners

In Sections 4.1 and 4.2, corners were presented as the simplest intermediate features which begin to combine line segments into more useful representations. By virtue of the vanishing point interpretations, it is even possible to tell which of the two primitives might be responsible for a particular corner. For example, a corner with arms labeled **h1** and **v** must be interpreted as part of a vertical wall on a rectangular primitive. However, corners still have positional ambiguities; in this example, the labeling information alone is insufficient to determine whether the corner is at the top of the rectangular primitive or the bottom.

The next level in the feature hierarchy bridges the gap between corners and the rectangular and triangular primitives, by partially modeling faces of the primitives. These features, called *2-corners*, result from the combination of two corners which share a common arm, and which lie on the same visible face of a primitive. A 2-corner can also be treated as a chain of three labeled line segments. By using 2-corners to model visible faces of the primitive polyhedra, it follows that a legal 2-corner must lie in a plane in object space. Under this condition, the vanishing point labels of the center segment in a 2-corner and one of the arms constrains the set of legal labels for the other arm, in much the same way that knowing the vanishing point label for one arm of a corner constrains the possibilities for the other arm. The vanishing point constraints for 2-corners will be described shortly.

Generation of 2-corners can be performed efficiently by constructing two lists for each line segment in the image, one list for each endpoint of the line segment. Each list enumerates the set of corners which have the given line segment as one arm, and form a corner at that endpoint. Then, if the lists have m and n corners respectively, $m \times n$ 2-corners can be generated by pairing each corner in one list with a corner from the other. As in corner generation, this multiplicative effect can produce large numbers of interpretations for 2-corners, in the absence of geometric constraints.

Figure 4-12 shows the 2-corners generated for the example Fort Hood area used in Section 4.2. There are 12 2-corners, with 90 interpretations (7.5 interpretations per 2-corner). Since many of the 2-corners overlap in the image, it is useful to illustrate the 2-corners individually. Figure 4-13 shows each 2-corner from Figure 4-12, with the number of interpretations indicated below each 2-corner.

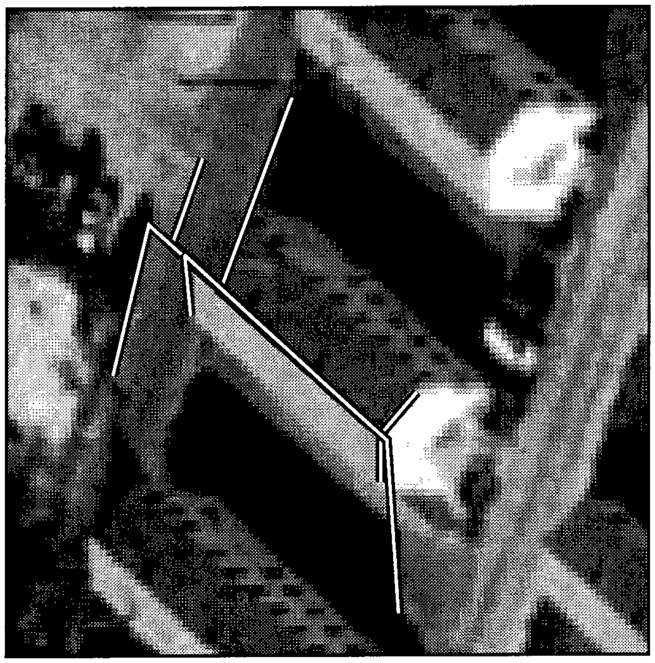


Figure 4-12: Initial set of 2-corners

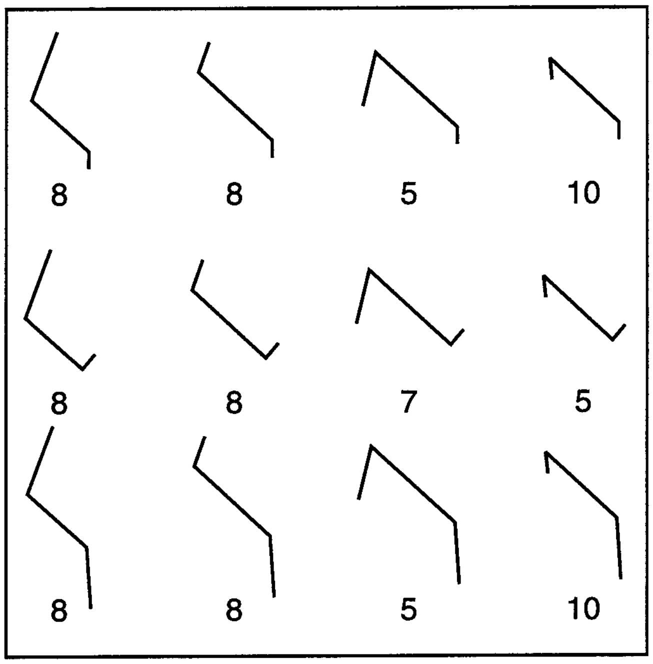


Figure 4-13: Number of interpretations for each 2-corner

By restricting a 2-corner's edges to lie on the same polygonal facet of a primitive, geometric constraints can again be employed to prune away illegal interpretations. These constraints also make use of the vanishing point line labelings, and hinge on the observation that, by definition, each edge *e* of a polyhedron is shared by exactly two polygonal facets, *A* and *B* [87]. There must therefore be a line segment at each endpoint of *e* which is part of *A*; the same is true of *B*. Moreover, the line segments associated with *A* must lie in a different plane than those of *B*; otherwise, the polyhedron would be ill-formed. So, if we attempt to form a 2-corner with *e* as the central edge and a line segment of *A* at one end, then we need not consider the line segment of *B* at the other end, since it lies in a different plane and cannot be part of the same polygonal facet.

There are three classes of geometric constraints on 2-corners; those that derive from rectangular primitives, those that derive from triangular primitives, and those that derive from both. In the diagrams that follow, the thick solid edge is the central edge of a possible 2-corner. The thin solid edge is a corner-forming edge which has been selected at one endpoint of the central edge. The thin dotted line(s) at the other endpoint represent the legal possibilities for the third edge of the 2-corner, given the first two. **v** represents a vertical, and $\mathbf{h}_a$ and $\mathbf{h}_b$ represent orthogonal horizontals. $\mathbf{p}_a$ and $\mathbf{p}_b$ represent slant orientations for triangular primitives. The two triangular facets of a triangular primitive are always formed by two slant line segments and one horizontal line segment. Since the horizontal line segment has the same orientation in both triangular facets, we will let this orientation be $\mathbf{h}_a$ without loss of generality.

Figure 4-14 shows the 2-corner constraints which apply to both primitives. Both constraints are essentially identical; when one horizontal forms a corner with an orthogonal horizontal, then the third arm of the 2-corner must also be orthogonal to the central edge. No other interpretation is legal, given the two primitives. These cases correspond to the "rooftops" or "floors" of the primitives.

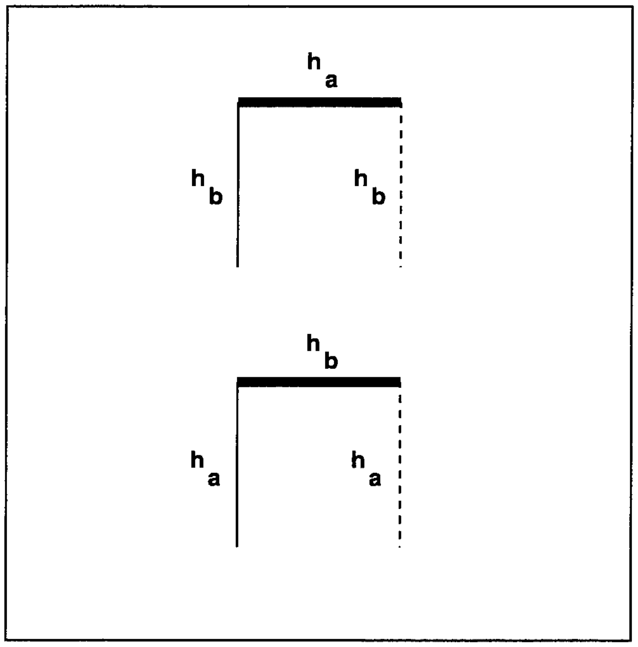


Figure 4-14: 2-corner constraints, both primitives

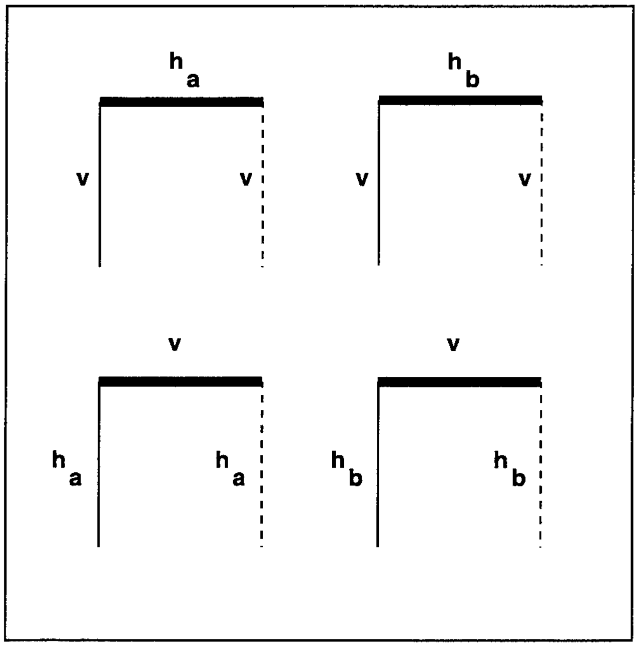


Figure 4-15: 2-corner constraints, rectangular primitives

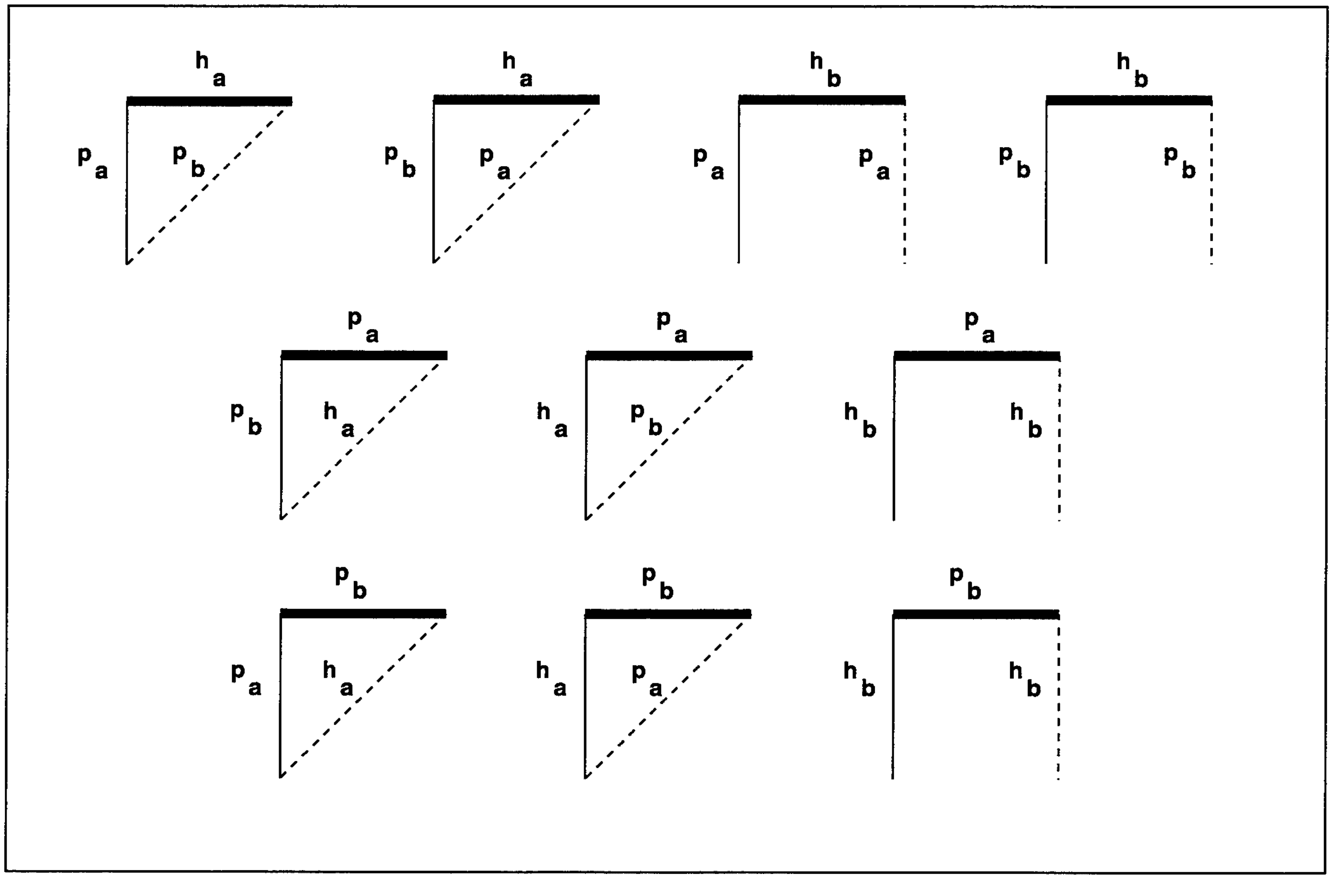


Figure 4-16: 2-corner constraints, triangular primitives

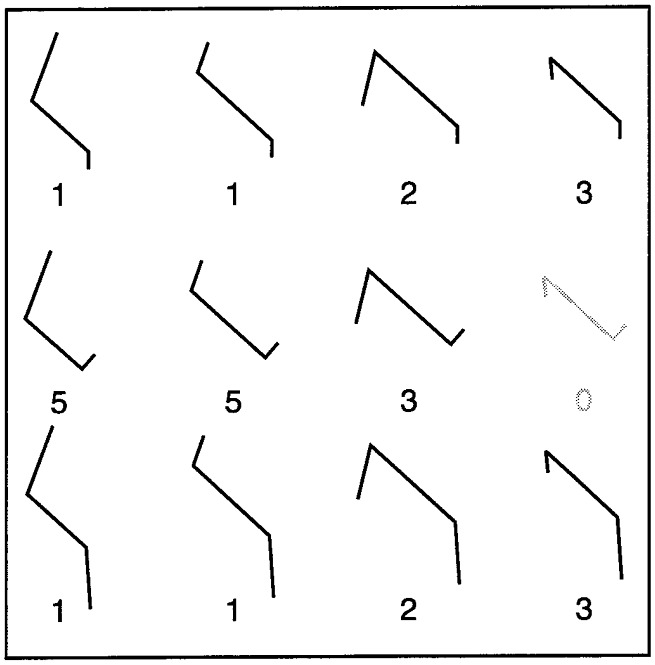


Figure 4-17: 2-corners after vanishing point constraints

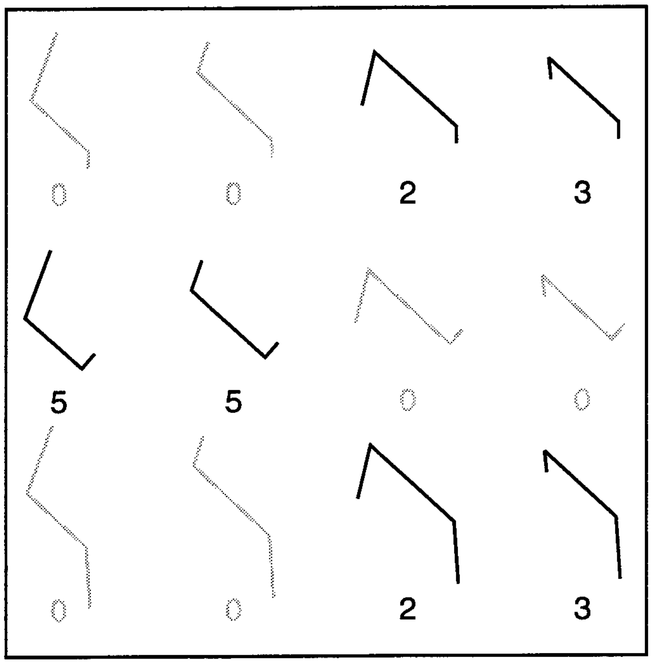


Figure 4-18: 2-corners after vanishing point constraints and convexity testing

Figure 4-15 shows the 2-corner constraints which are specific to the rectangular primitive. In the top two cases, if a horizontal is the central edge and a vertical forms a corner at one end of this edge, then the other arm must also be vertical. In the bottom two cases, if a vertical is the central edge, and a horizontal forms a corner at one end, then the other arm must share that same horizontal orientation. All four situations correspond to the sides of the rectangular primitive.

Figure 4-16 shows the 2-corner constraints which are specific to the triangular primitive. In all of these cases, at least one of the first two edges has a slant orientation. Six of the cases correspond to the triangular facets; these cases are readily identified since the first two edges are both peaks, or one of the edges has orientation $\mathbf{h}_a$. These six cases are essentially the same; given two edges with known orientations, the third orientation is uniquely determined, since the three edges are known to form a triangular facet. The remaining four cases correspond to the rectangular sides of the triangular primitive.

These constraints can easily be applied during the 2-corner generation algorithm described earlier. In that algorithm, a list of corners was assembled for each endpoint of an edge, where each list contained m and n corners respectively. Each corner in the first list was paired with each corner in the second list to generate $m \times n$ 2-corners. Now, when a corner is selected from the first list, each corner from the second list can be tested to see if its combination with the other corner would result in a legal 2-corner. If the combination is illegal according to the constraints, then no 2-corner is produced for this pairing.

Figure 4-17 shows the result of applying these vanishing point constraints on the 2-corners depicted earlier in Figure 4-13. One 2-corner is eliminated completely, since all of its interpretations are illegal with respect to the vanishing point constraints for 2-corners. The remaining 2-corners show significant reductions in the number of interpretations; 11 corners are left with 27 interpretations, averaging 2.5 interpretations per 2-corner.

There are still more geometric constraints available for 2-corners, the mathematics of which are outlined in Section A.6. Since 2-corners are assumed to be part of a primitive face, and since each face of the primitives is convex, it follows that each legal 2-corner must be convex. This can be easily implemented by a pair of 2D determinant tests, as in Equation (A.25), to verify that the 2-corner "turns" in the same direction at each corner point. The convexity tests are strict, since two line segments of the same orientation never form a legal corner, as described in Section 4.2. Figure 4-18 shows the result of applying convexity tests to the 2-corners from Figure 4-17. Several 2-corners are completely eliminated by this test, leaving 6 2-corners with 20 interpretations (an average of 3.3 interpretations per 2-corner). Just as importantly, the desired 2-corner (in the upper right hand corner of the figure) is still present, with 3 legal interpretations.

This running example does not illustrate any triangular facets, which have additional geometric constraints. Since 2-corners are convex, either interior angle α of the 2-corner need only obey $\alpha<\pi$. Triangles must obey the more strict interior angle constraint of $\alpha<\pi/2$ for at least two of their interior angles. Determinant tests can once again be employed to verify that this constraint holds for a potential 2-corner triangle; Equation (A.26) is used for this situation.

One more constraint is available for triangular facet 2-corners; a test for checking the relative distance of the intersection point of the arms of the 2-corner to the 2-corner itself. One possible approach would be to examine the corner data produced by the range search and see if the two arms formed a legal corner. However, if noise or occlusions interfered with that corner, then this approach would fail even though the triangular facet might be valid. A better approach is to use a relative distance constraint. The minimum and maximum x and y coordinates of the 2-corner are computed to form a bounding box; these minimum and maximum values are then extended by the length l of the longest line segment in the 2-corner, creating a bounding box which has been expanded by l pixels on each side. If the intersection point of the arms falls outside of this box, the triangular interpretation is rejected.

Figure 4-19 shows the final set of 2-corners which were generated for the larger Fort Hood image in Figure 4-1. As with the corner example in the previous section, Table 4-2 summarizes the reduction in the number of 2-corners and interpretations as geometric constraints are added to the 2-corner generation module. Vanishing point constraints and convexity testing provide the most significant reduction in the number of interpretations and 2-corners, respectively. The triangle-based constraints do not provide as substantial an impact here, since triangular 2-corners occurred infrequently in the Fort Hood image. Most of the barracks buildings do not have an edge where the triangular prism meets the rectangular volume, and so triangular 2-corners were not generated.

As in the corner generation phase, geometric constraints provide leverage for attacking the combinatorial difficulties of 2-corner generation. The combination of vanishing point constraints, which specify legal geometries with respect to the primitives, and convexity constraints, which specify legal geometries under restriction to planar facets, significantly reduces the search space for primitives. Since more processing power is expended on intermediate features as they increase in complexity, the benefits of following Principle 3 are clear.

2-corners combine three line segments to form an intermediate feature; it is worth noting that they are not unique in this respect. Another popular feature is the *trihedral vertex*, which consists of three line segments which meet at a single corner point. Much research has been done on the use of this feature for intermediate vision, and it was also the subject of early work on line drawing interpretation [17, 43, 48, 102]. More recently, orthogonal trihedral vertices have been used to model rectilinear structures in aerial images [55].

The difficulty with trihedral vertices is that they may not always be visible from certain viewpoints. For example, neither the rectangular or triangular primitives have any visible trihedral vertices when viewed from directly above. More generally, when either primitive is viewed from an angle which is perpendicular to one of its faces, no trihedral will be visible for that primitive. 2-corners do not suffer from this problem, since at least one face of a polyhedron must be visible from any viewpoint, barring occlusions. Since the face is an n-sided polygon, there are n possible 2-corners for this polygon. Of course, occlusions and edge fragmentation can prevent the detection of 2-corners, but the same is true of trihedral vertices.

For this work, 2-corners will be used exclusively, but this should not be taken as a total rejection of trihedral vertices as useful intermediate features. It is easy to imagine oblique views of objects where edge fragmentation is sufficiently severe that no 2-corners can be detected, and only trihedral vertices can be detected. The best approach, left to future work, is to employ both intermediate features. This topic is reiterated in Chapter 7.

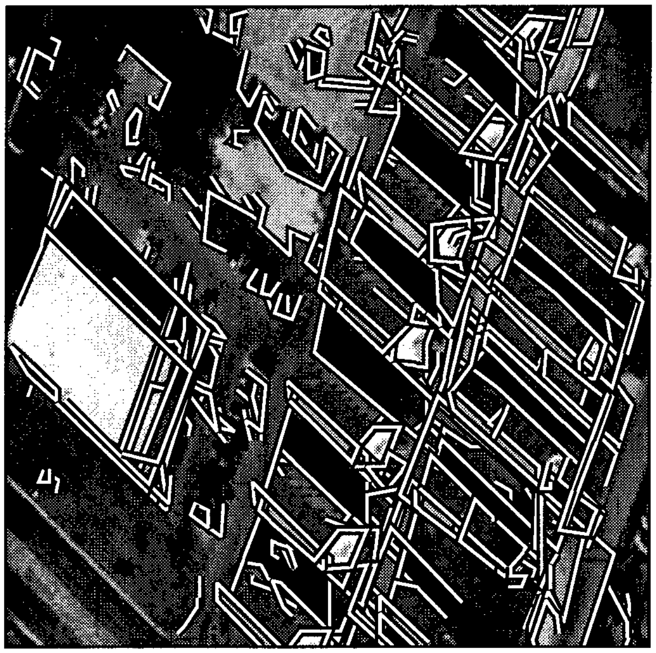


Figure 4-19: Final set of 2-corners

constraints	# of 2-corners	# of interpretations
none	3460	24740
vanishing point constraints	2217	8378
vanishing point constraints + convexity tests	1193	4654
vanishing point constraints + convexity tests + interior angle constraint	1148	4536
vanishing point constraints + convexity tests + interior angle constraint + relative distance test	1141	4498

Table 4-2: Effects of geometric constraints on 2-corners

4.4. Generating primitives from intermediate features

In the previous section, the correspondence between 2-corners and facets of the primitives was used to geometrically constrain the interpretations of the 2-corners. In this section, the correspondence is pushed further; for the primitives used in this work, the mapping from a 2-corner under a specific interpretation to one of the two primitives is unique. This allows PIVOT to instantiate primitives directly from 2-corners, by a combination of geometric analysis and search.

The first phase of this process establishes a mapping from a 2-corner to a primitive. There are two components to this process; establishing whether the 2-corner belongs to a rectangular primitive or a triangular primitive, and determining the relative object space positioning of the 2-corner with respect to the ground. The second phase instantiates a partial primitive using the information from the first phase, and executes a search to complete empty edge slots in the primitive. Finally, subsets of the edge data, which have differing extents in the image, are used to hypothesize complete primitives in the image, and an image analysis is performed to select the completed primitives which are best supported by the underlying edge data.

This process completes the intermediate feature hierarchy, and gives PIVOT a set of geometrically consistent primitive hypotheses which are best supported by the image data. As in previous steps up the feature hierarchy, the use of geometric constraints provided by vanishing point labeling and analysis plays a key role in efficiently generating more complex features.

4.4.1. Mapping 2-corners to primitives

The first step in moving from 2-corner representations to primitive instances involves determining whether a given 2-corner belongs to a rectangular volume primitive, or a triangular prism primitive. For brevity, the rectangular volume primitive will henceforth be referred to as a `rect`, and the triangular prism primitive will be referred to as a `peak`. The algorithm for deciding whether a 2-corner is a `rect` or a `peak` is quite simple, and is based on the vanishing point labels of the 2-corner:

- if the central edge of the 2-corner has label **v**, then the 2-corner belongs to a `rect`;
- else, if the central edge of the 2-corner has label **p**, then the 2-corner belongs to a `peak`;
- else, if the central edge of the 2-corner has label **h**, then
    - if the arms of the 2-corner are labeled **v**, then the 2-corner belongs to a `rect`;
    - else, if the arms of the 2-corner are labeled **p**, then the 2-corner belongs to a `peak`;
    - else, if the arms of the 2-corner are labeled **h**, then the 2-corner belongs to a `rect`

An inspection of Figures 3-1 and 3-2 shows that these tests map 2-corners to primitives. One possible point of confusion may arise when one considers a 2-corner for which all segments have orthogonal **h** labels. This 2-corner would appear to belong to both primitives at first glance. Recall from Section 4.3, however, that a 2-corner is assumed to belong to a visible face of a primitive, and from Section 3.2 that the primitives are assumed to sit on the ground. Since the only mappings of this 2-corner to a `peak` all belong to the ground face of the `peak`, which can never be seen by the camera, the 2-corner must be assigned to a `rect`.

Since this mapping of 2-corners to primitives is unique, we can ignore the issues which arise when multiple primitives are used. The labeling-to-primitive tests may not always result in a 1-to-1 mapping of 2-corner interpretations to primitives if a larger set of primitives is used, particularly when primitives share faces with similar labelings. For example, if we used a third primitive which had a non-ground rectangular face, it would have been necessary to map a 2-corner with an **h1–h2–h1** interpretation to both the `rect` and this new primitive. For a large set of primitives which share many legal 2-corner assignments, this could lead to another potential combinatorial explosion in the search space of primitive instances. Perhaps the best approach for alleviating this problem would be to introduce new intermediate features, more complex than 2-corners, but less complex than a complete instantiation of a primitive. Geometric analysis similar to that performed in Sections 4.2 and 4.3 could then be used to constrain the search space to a manageable level.

The tests above establish whether a 2-corner belongs to a `rect` or to a `peak`. They do not establish, however, exactly to which edges and points of a primitive a 2-corner should be assigned. For example, consider a 2-corner with the labeling **h1–v–h1**. By the tests above, we know that it must belong to a `rect`. What we do not know is whether the first segment labeled **h1** lies on the ground or is part of the roof face. Neither do we know whether the other **h1** segment lies on the ground or the roof. Alternatively, the problem is that we do not know which endpoints of the interior **v** segment of the 2-corner correspond to the ground and roof.

Fortunately, this information can be easily computed by vanishing point analysis. For any line segment with a non-horizontal vanishing point label, the computation can be carried out by determining which endpoint of the line segment is closer to the image plane representation of the line segment's vanishing point. Let $<x_q,y_q>$ be the image coordinates of the vanishing point, and let $\mathbf{q}$ be the vector representation of the vanishing point (see Section A.3 for a quick refresher on vanishing point mathematics). Using Equation (A.13), we compute $\mathbf{v}$, the object space orientation vector corresponding to $\mathbf{q}$. Letting D be the Euclidean distance function, and letting $<x_1,y_1>$ and $<x_2,y_2>$ be the endpoints of the line segment, we have the following algorithm:

```
• Let $d_1 = D(<x_1,y_1>,<x_q,y_q>)$
• Let $d_2 = D(<x_2,y_2>,<x_q,y_q>)$
• If $\mathbf{v}_z > 0$ then
    • If $d_1 < d_2$ then point 1 is closer to ground
    • Else if $d_1 > d_2$ then point 2 is closer to ground
• Else if $\mathbf{v}_z < 0$ then
    • If $d_1 < d_2$ then point 2 is closer to ground
    • Else if $d_1 > d_2$ then point 1 is closer to ground
```

For any 2-corner with a non-horizontal line segment, this algorithm can be used to establish the relative object space positions of its points with respect to the ground, allowing an assignment of the line segments and points that make up a 2-corner to specific slots in either a `rect` or `peak` primitive.

For 2-corners which correspond to triangular facets of a `peak`, one more constraint can be applied after this vanishing point analysis. Since we expect primitives to sit on the ground, we also expect the triangles of `peak` primitives to point skyward. Hence, if the vanishing point analysis indicates that the points joining the **h** segments to the **p** segments are further away from the ground than the opposite endpoints of the **p** segments, the triangle is pointing towards the ground, and can be rejected.

4.4.2. Instantiating primitives in image space

At this stage of PIVOT's processing, enough information is available from vanishing point analysis to create a primitive "placeholder" data structure. This structure defines the topology for each primitive, and has fields for each edge and corner point in the primitive. Each edge field contains slots for a vanishing point label and a pointer to a line segment from the original line extraction data. Each 2-corner is used to fill three of these edge slots. While the remaining edge fields do not yet have pointers to line segments, their vanishing point labels can be filled in, since the 2-corner's labels uniquely determine the labels for all edges in the primitive.

Figure 4-20 shows the assignment of a 2-corner to a placeholder `rect`, as well as the labeling of the edges in the primitive based on the 2-corner's labels. In this example, the 2-corner has vanishing point labels **v–h2–v**, and vanishing point analysis has determined that the horizontal must lie on the roof, so the 2-corner is inserted in the appropriate position. Recall that since orthogonal horizontal vanishing points are found in pairs by the vanishing point detection techniques of Section 3.4, the horizontal edges perpendicular to the 2-corner must be labeled with **h1**, the orthogonal partner of **h2**.

Figure 4-21 shows the assignment of a 2-corner to a placeholder `peak`. In this example, the 2-corner has vanishing point labels **h2–p1–h2**. Vanishing point analysis shows that for this 2-corner, the first **h2** must lie on the ground and the second must lie on the peak ridge. Again, since orthogonal horizontal vanishing points are found in pairs, the perpendicular horizontal edges in the `peak` must be labeled **h1**; similarly, since the slanted peak vanishing points are also found in pairs, the slanted edges on the opposite side of the `peak` must be labeled **p2**.

To solve for the image space coordinates of every corner point of each placeholder primitive, it is necessary to locate edges to fill the empty edge slots of each primitive. Of course, due to edge fragmentation, occlusions, and image noise, it is unlikely that every visible edge slot can be filled. However, for any given placeholder primitive, it is only necessary to find enough edges so that the extent of each dimension of the primitive is determined.

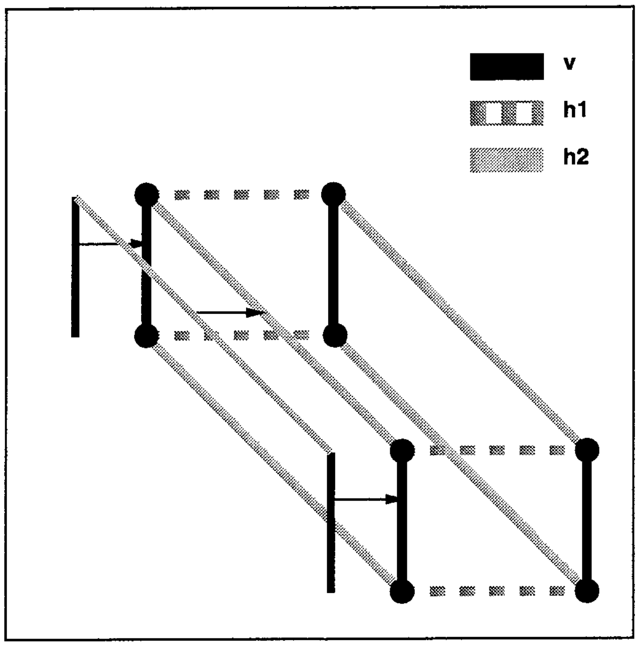


Figure 4-20: Creating a rectangular placeholder

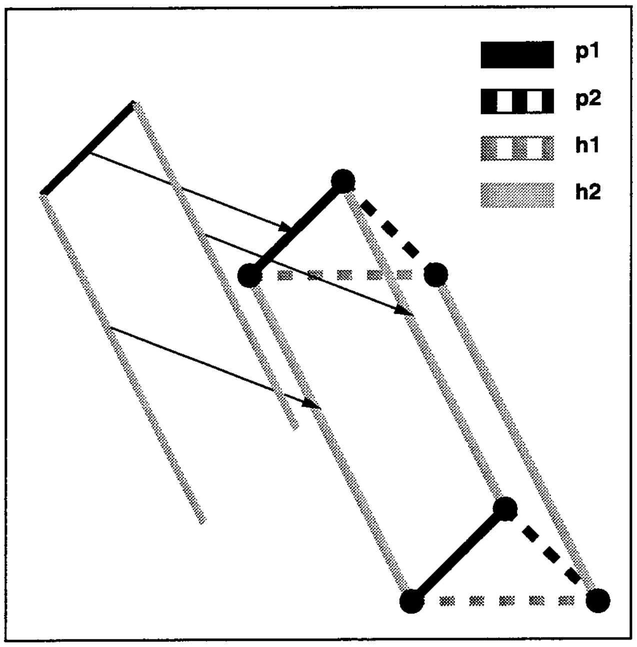


Figure 4-21: Creating a triangular placeholder

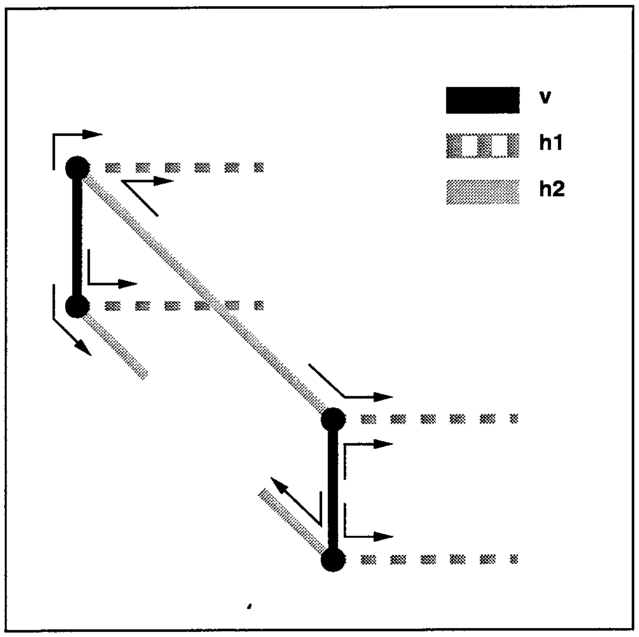


Figure 4-22: Searching for corners

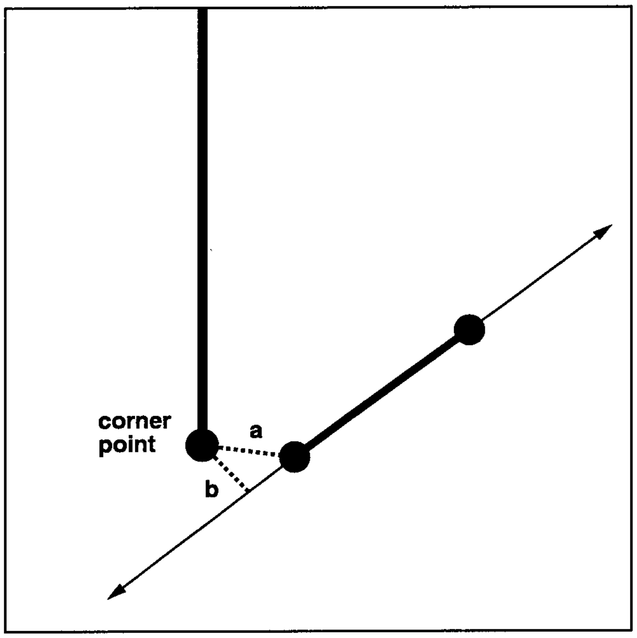


Figure 4-23: Local distance test for edge selection

For example, a `rect` can be parameterized in terms of three dimensions; length, width, and height. If enough edges can be found to provide measurements of each of these dimensions, then enough information would be available to solve for the coordinates of each corner point of the primitive. A `peak` can be parameterized in terms of only two dimensions; length along the horizontal, and the length of any side of a triangular face. This is a consequence of the geometry of triangular faces under projection; if the coordinates of one edge (and hence two points) of a triangle are known, then the vanishing lines passing through these endpoints intersect at the third point of the triangle.

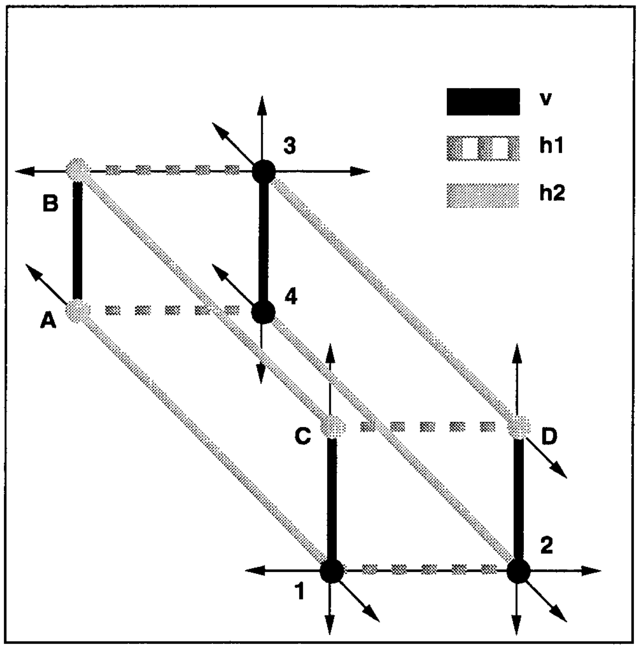


Figure 4-24: Solving for point positions in a rectangular primitive

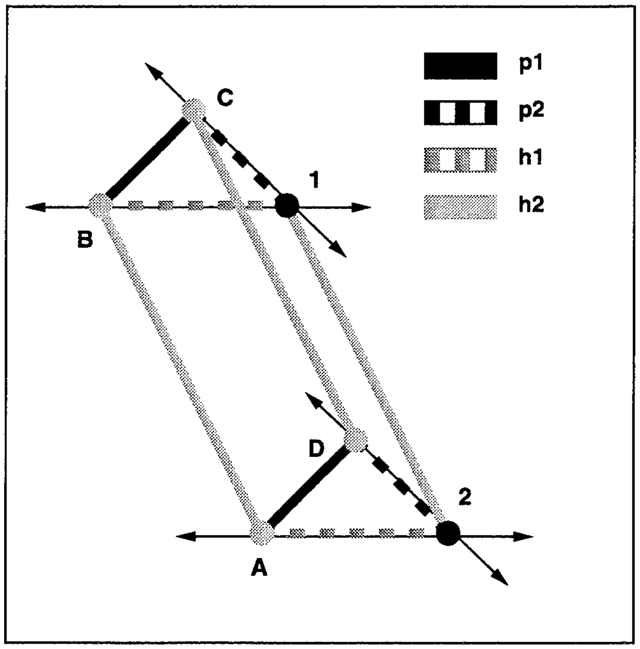


Figure 4-25: Solving for point positions in a triangular primitive

It should be noted, of course, that even if no edge fragmentation, occlusions, or image noise are present, it may still be impossible to obtain measurements in each dimension solely by locating object edges. For example, if an image was acquired from directly above a scene consisting of a rectangular building, none of the vertical line segments in the building would be visible in the image. In this case, other image features would be necessary to infer height; Chapter 5 describes the use of shadow analysis to handle these cases.

Nonetheless, if we seek measurements of each independent dimension for each primitive, then a simple approach can be taken. At each known corner point in the placeholder primitive, a search is performed on the corner data generated by the algorithms of Section 4.1, to find corners which share one arm with a line segment of the 2-corner, and which fill in an empty slot of the placeholder with the other arm, such that the vanishing point labeling of this arm is consistent with the label of the empty slot in the placeholder. This search is carried out for each `rect` placeholder; it only needs to be carried out on `peak` placeholders when the 2-corner fills a triangular facet, since otherwise both dimensions of the `peak` are known. Figure 4-22 graphically depicts such a search for the 2-corner example in Figure 4-20. Each arrow is aligned with an edge from the 2-corner, and an empty slot in the placeholder which needs to be filled. Each arrow points towards the direction in which the corner must lie to form a consistent primitive with the 2-corner.

This search can potentially generate many possible edges for a single edge slot in a placeholder primitive, since there may be multiple corners which satisfy each arrow in Figure 4-22. For slots which contain multiple edges after the search, a local distance test is used to select the edge which has the best alignment with its connecting corner point. Figure 4-23 illustrates the test. For every edge in a slot, the distance from the corner point to the nearest endpoint of an edge (*a* in the diagram) and the perpendicular distance from the corner point to the extended line formed by the edge (*b*) are computed. The edge with the smallest value of $a+b$ is selected, and the rest are discarded. It should be noted that this test violates Principle 4, since it is based on local information.

When each edge slot has been processed to have at most one edge, then primitives are computed by selecting subsets of the filled edge slots and performing vanishing line intersections to solve for the positions of the unknown points. This is an iterative process, in which the position of an unknown point is solved as soon as two adjacent points on the primitive are known. Figures 4-24 and 4-25 illustrate this solution process for three particular edges for the `rect` and `peak` placeholders, respectively.

In Figure 4-24, the points whose positions are known are shaded grey and labeled with letters. The points whose positions are initially unknown are colored black. The algorithm begins by locating a point which is adjacent to two known points; point 1 is such a point, adjacent to A and C. Its position is computed by intersecting the **h2** vanishing line through A with the **v** vanishing line through C. Point 2 is then adjacent to two known points, 1 and C, and its position is solved next by intersecting the **h1** vanishing line through 1 with the **v** vanishing line through D. Point 3 can be solved by intersecting the **h1** vanishing line through B with the **h2** vanishing line through D, and point 4 is solved by intersecting the **v** vanishing line through 3 with the **h2** vanishing line through 2.

The solution for the `peak` placeholder is simpler, since in this case there are fewer unknown points. Point 1 can be solved by intersecting the **h1** vanishing line through B with the **p2** vanishing line through C, and point 2 can be determined by intersecting the **h1** vanishing line through A with the **p2** vanishing line through D.

If a placeholder does not have enough edge slots filled to completely solve for every point position (i.e., only one face of a primitive is visible in the image), the algorithm solves for as many point positions as possible. Also, since this algorithm tries all minimal subsets of filled edges to generate completed primitives, each placeholder can generate multiple solutions.

Finally, for each solution, the parallel edges of the primitives are tested to ensure that they do not intersect in image space, and each edge is tested to ensure that it lies on or near its vanishing line. In PIVOT, an edge is allowed to lie within 15 degrees of a slanted peak vanishing line, and within 7.5 degrees for horizontal and vertical vanishing lines. This distinction is made since slanted peak vanishing lines are detected with less accuracy than their horizontal counterparts, as described in Section 3.6. To avoid confusion, it should be noted that all of these solutions are comprised of 2D image space point coordinates; no 3D information has been computed yet.

Before continuing, it is worthwhile to examine the advantages and shortcomings of this particular approach to solving for the image space positions of primitive points. This algorithm is fast, since it only involves a small number of line intersections in image space; it also correctly handles perspective, since it intersects vanishing lines in image space. It also has the advantage that fragmented edge data can be ignored, if good edges exist for each dimension of the primitive, since each minimal subset of edges is used for the vanishing line intersection solution process.

A potential disadvantage is that the solutions are order-dependent; the order in which unknown points are solved can affect the positions of their solutions, since different vanishing lines and points would be used for the intersection computations. Another potential disadvantage is that if no good edges can be found for a particular primitive dimension, the resulting primitive solution will be poorly delineated. For example, if only one vertical edge is found for a `rect` and it is truncated, then the resulting completed primitives will also be truncated in height. An alternate solution method which was not explored in this work is a constrained least squares solution, which would take all edges as input and produce a unique solution for the point positions. The advantage of this approach is that it could handle all of the edge data simultaneously, subject to the shape constraints of each primitive, and eliminate the need for heuristic tests to evaluate the fit of the primitive to the vanishing lines. The potential disadvantage of a least squares solution is speed, compared to the iterative approach just described. The development of an efficient constrained least squares solution method for the vanishing line primitive completion problem is left to future work.

Figure 4-26 revisits the running example from Figure 4-18. For each 2-corner, the number of interpretations is given (as in Figure 4-18). The number of vanishing line intersection solutions for each 2-corner is also given, as well as the number of solutions remaining after heuristics are applied to check the fit of the solutions to the vanishing lines. Initially, 40 solutions were generated from these 6 2-corners; only 12 remained after the heuristic tests. Figure 4-27 shows the 12 primitive solutions which remained after the heuristic tests. The 2-corner that generated each solution is overlaid on top of the primitive. Each figure is also marked with the letter used to identify the 2-corner in Figure 4-26.

Figure 4-27 illustrates several aspects of the primitive solution process. First, notice that different interpretations of a 2-corner can lead to very different solutions for primitive point positions. This is best exemplified by the three solutions resulting from 2-corner *(c)*. One solution, *(c1)*, is an incomplete attempt to find a `rect` when no verticals were present at the corner points; *(c2)* is a `peak` primitive aligned along one horizontal direction; and *(c3)* is another `peak`, this time aligned along the orthogonal horizontal direction.

The solutions for *(a)* and *(c2)* illustrate another aspect of the vanishing line intersection method. Since the method looks at minimal subsets of the edges, it can eventually find a solution which ignores edges with incorrect extents. The solution in *(a)* uses the central segment and the short arm of the 2-corner, and all of the intersection computations are based on the three points which define these two segments. The fourth point was not used for this particular solution, since only two edges were needed to completely solve for the positions of the remaining points. In the other solution for this 2-corner with the same interpretation, the central segment and the long arm were used, but this solution was rejected by the heuristics, since its horizontals were in poor alignment with the horizontal vanishing line. A similar description applies to the solution in *(c2)*.

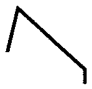
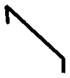
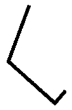
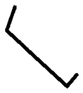
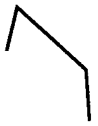
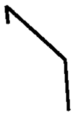
						
	(a)	(b)	(c)	(d)	(e)	(f)
# of interpretations:	2	3	5	5	2	3
# of initial solutions:	4	6	10	10	4	6
# of consistent solutions:	1	2	3	3	1	2

Figure 4-26: Number of vanishing line intersection solutions for 2-corners

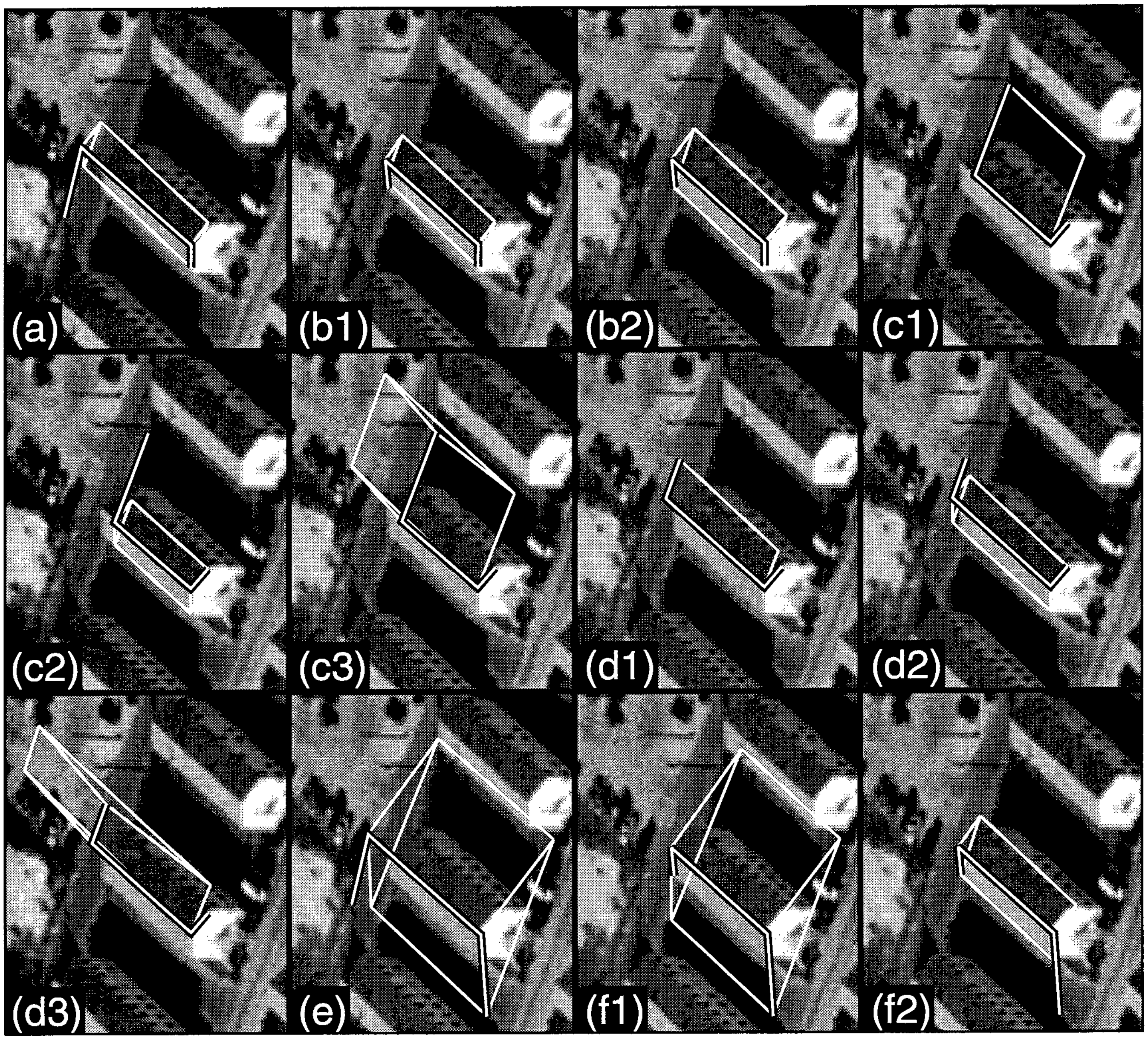


Figure 4-27: Image space primitive solutions for 2-corners

The solutions for *(b1)* and *(b2)* highlight this aspect of the method from a different angle. Both solutions used the same interpretation for the 2-corner; note, however, that the *(b2)* solution is slightly larger than the *(b1)* solution. Again, this is a consequence of using different minimal sets of edges to solve for point positions. In this case, the slanted peak edges used in the solutions had slightly different orientations and extents in the image, and hence the point positions computed by intersection were slightly different as well.

Of course, this hypothesis generation method implicitly assumes that for each extent of a primitive volume, at least one line segment correctly measures it. In practice, occlusions and noise often result in truncated line segments for a particular edge of a primitive, which results in a partial volumetric description. This problem is addressed in detail in Sections 5.1 and 5.4, where methods for combining and extending primitives are presented for handling primitive fragmentation.

4.4.3. Selecting a unique primitive for each 2-corner

The final stage in mapping 2-corners to primitives involves the selection of a single primitive for each 2-corner. Since each 2-corner is intended to correspond to one planar face of a polyhedron, it follows that each 2-corner can belong to at most one primitive. This section describes the mechanisms by which one primitive is selected from potentially many primitive solutions for a 2-corner.

Perhaps the most difficult aspect of selecting one solution is that all remaining solutions are geometrically consistent with respect to the detected vanishing points. Each edge of each primitive lies on or near its expected vanishing line, since the solution process enforces this constraint. As a consequence, since the primitive geometries are consistent with the vanishing lines, and internally consistent in terms of forming realizable 3d volumes, no further constraints can be derived from vanishing point analysis. It follows that other tools are necessary to choose a best solution for each 2-corner.

Since the primitive geometries are legal, a natural recourse is to look for image support for each primitive solution. Certainly, if we examine Figure 4-27 once more, it is reasonable to expect that solution *(c2)* is a better fit to the image data than solution *(c3)*. The question is how to define "better fit" in a robust way, in the absence of purely geometric constraints. This question also arises naturally in the context of model-based vision, where a CAD model is given as input to a vision system, and the system must detect and localize instances of the model in an image.

In model-based systems, a recently popular approach involves the use of the Hausdorff distance metric for comparing models to images. Informally, if the Hausdorff distance between two 2D point sets A and B is d, then every point in A is at most d units away from some point in B, and vice versa. In this approach, the model points and the edge points in the image each form a point set. Matches are found when the Hausdorff distance between a model point set and the edge point set falls below a predefined threshold. A thorough discussion of implementation issues, including efficient handling of rotations and translations of the model point set, is presented elsewhere [44].

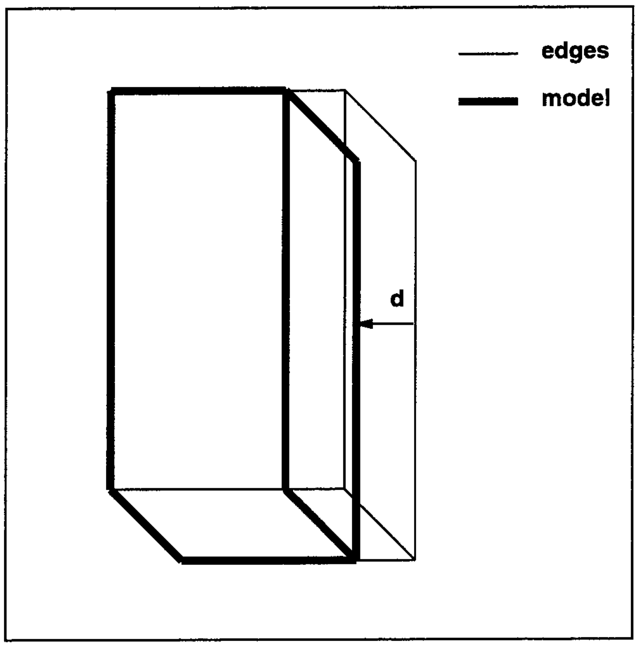


Figure 4-28: First model, Hausdorff distance *d*

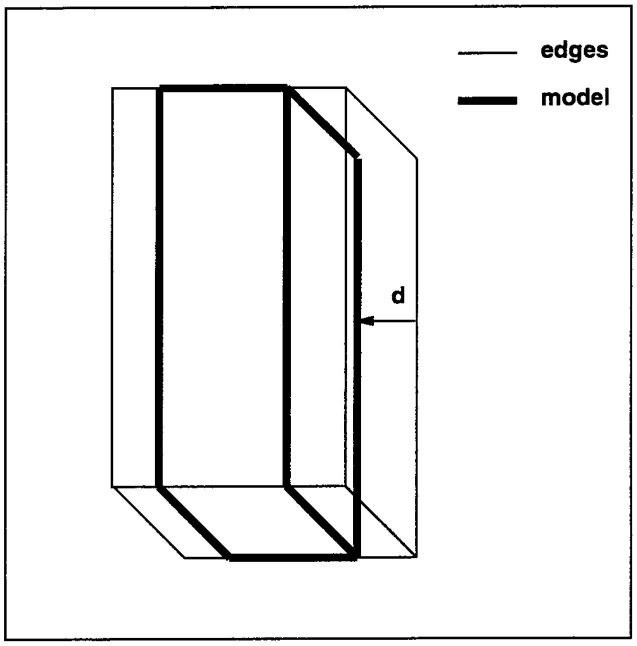


Figure 4-29: Second model, Hausdorff distance *d*

While the Hausdorff distance metric has desirable properties for model matching, and while it explicitly formalizes the notion of a "good fit" between a model and an image, it does not have sufficient power to be used as a discriminatory metric for hypothesis selection. Consider Figures 4-28 and 4-29, which depict two different model solutions overlaid on image edges.

In Figure 4-28, the Hausdorff distance is *d*, since the edge points at the base of the building are further away from the model than any other edge points, and since none of the model points are further than *d* units away from edge point. But the same is true of the model in Figure 4-29, even though its delineation of the building boundaries is worse. It should be clear that there exists a wide range of models which would have Hausdorff distance *d*, even though the quality of their delineations might differ significantly.

To solve this problem, an alternate metric is used. Given two point sets A and B, the function $\delta(A,B,\varepsilon)$ is defined to be the fraction of points in A which are within ε units of at least one point in B. Then, $\Delta(A,B,\varepsilon) = 0.5(\delta(A,B,\varepsilon)+\delta(B,A,\varepsilon))$. The Δ distance function measures the fit of image points to model points, and vice versa, within some predefined tolerance. This distance function shares similarities with the visible edge component of the verification function used in the SITECITY semi-automated modeling system [40]. For both PIVOT and SITECITY, ε is set to 2 pixels to allow for noise in the edge detection process and discretization error in line segments.

For fast computation of Δ, a chamfer distance transform [10] is applied once to the extracted edge data for the entire image. The points on the model are then used as indices into the distance transform image, to calculate how many points on the model are within ε pixels of edge points. This computes one term of Δ; to compute the other term, the minimum bounding rectangle (MBR) of all primitive solutions for a given 2-corner is computed, and each solution is individually chamfered within the bounds of the MBR. The edge points within this MBR are then used to index into each solution's distance transform image.

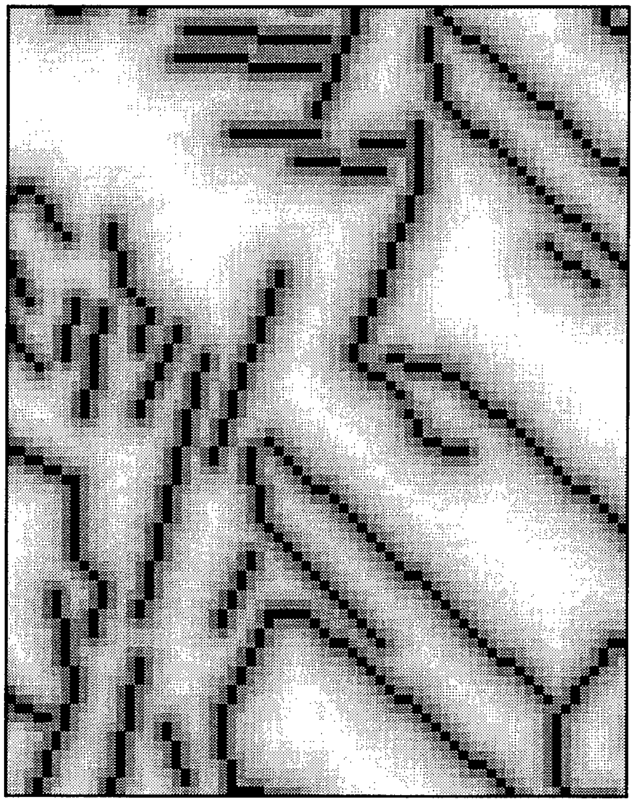


Figure 4-30: Chamfer of edge data for 2-corner *d*

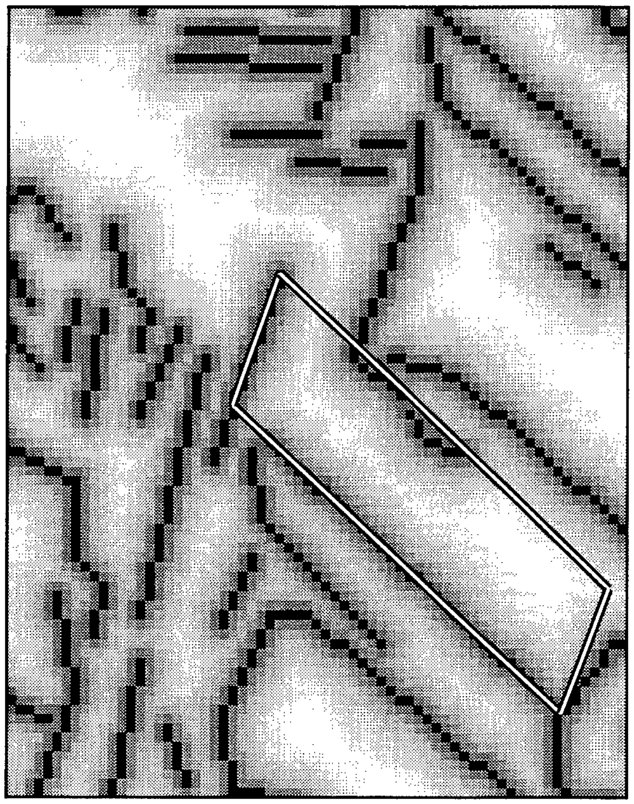


Figure 4-31: Solution *d1* on chamfer image

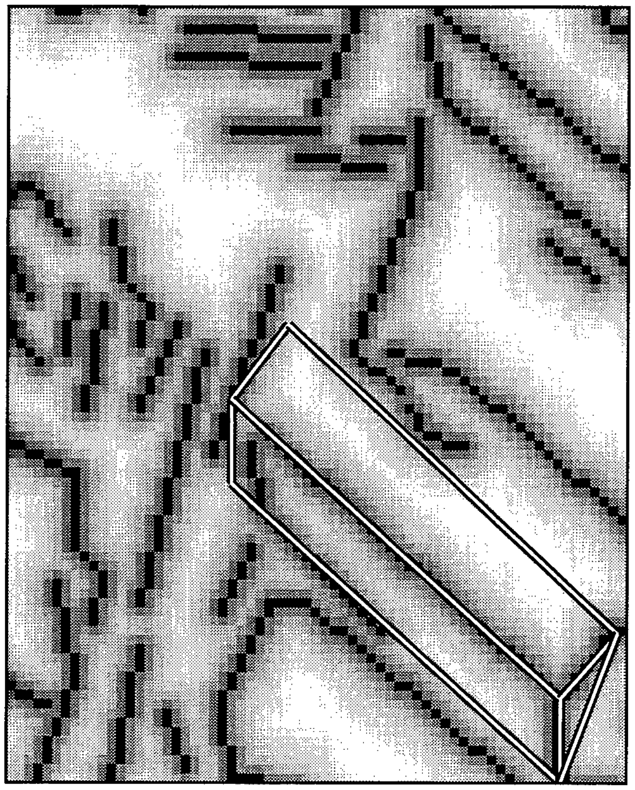


Figure 4-32: Solution *d2* on chamfer image

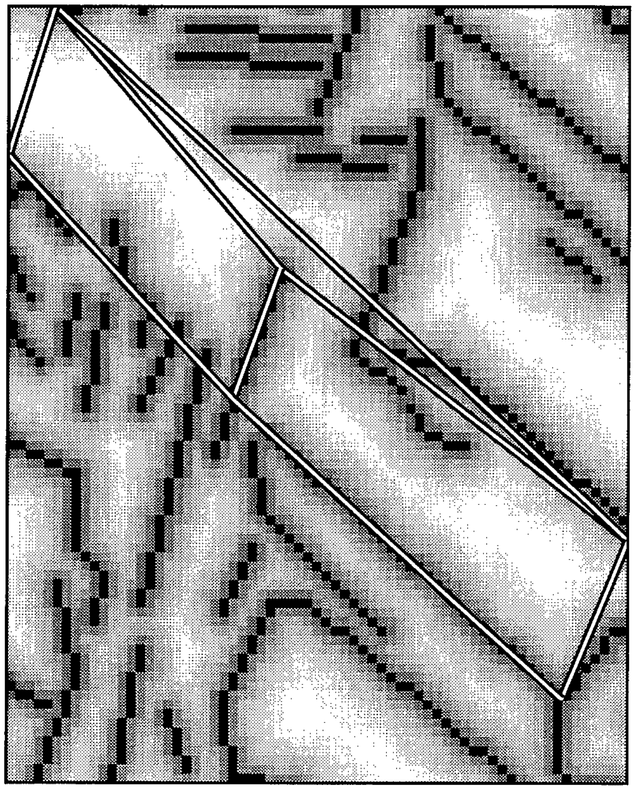


Figure 4-33: Solution *d3* on chamfer image

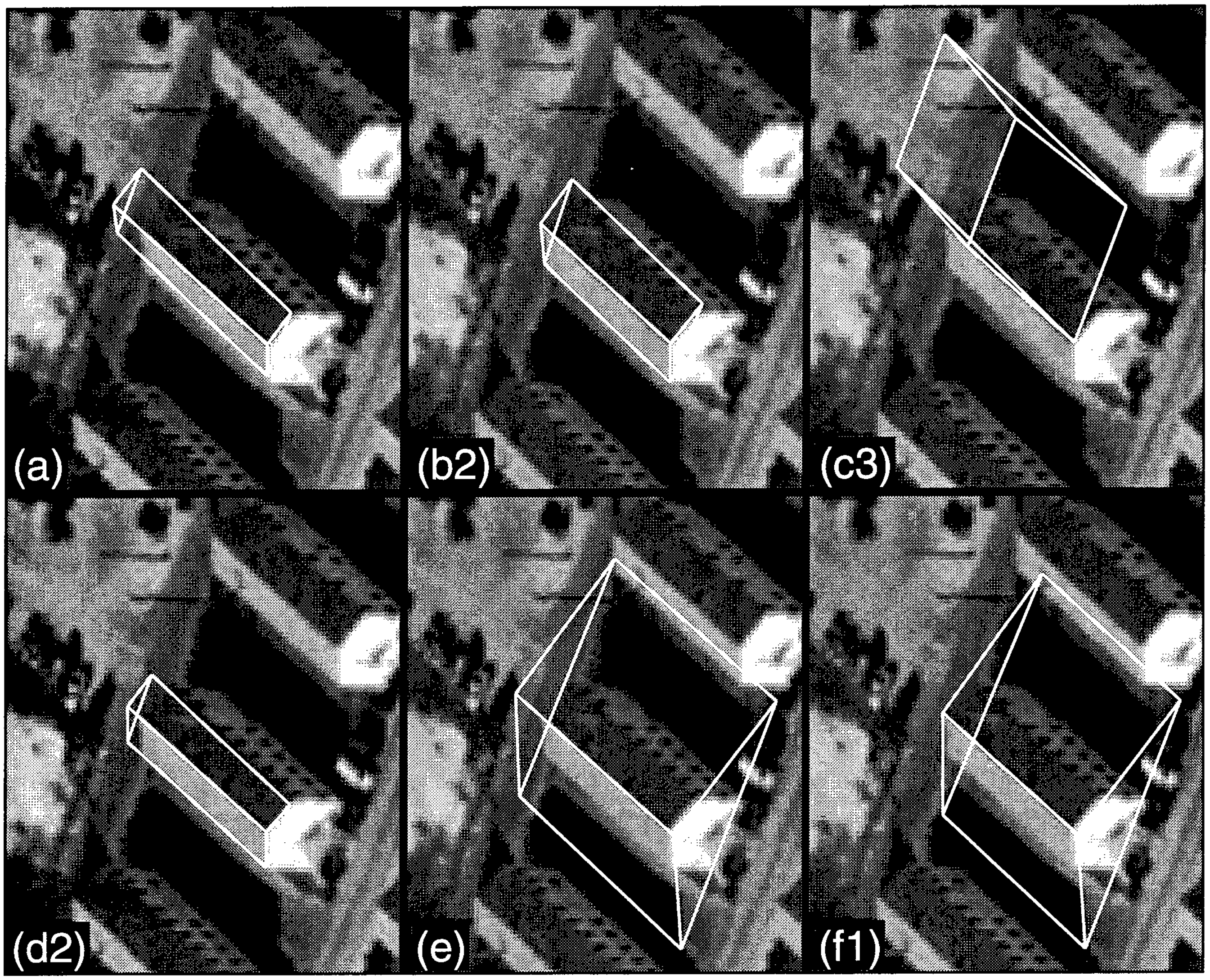


Figure 4-34: Selected image space primitive solutions for 2-corners

Figure 4-30 shows the distance transform for the edge data inside the MBR of the *(d1)*, *(d2)*, and *(d3)* solutions in Figure 4-27. Figures 4-31, 4-32, and 4-33 show each of these solutions overlaid on the distance transform images. In these images, black pixels correspond to edge points (zero distance) and brighter pixels encode increasing distance from the nearest edge point. The values of Δ for each of these solutions are 0.4247, 0.4249, and 0.3790, respectively, so in this case *(d2)* would be selected as the final primitive solution for 2-corner *d*. It is informative to examine each term of Δ for each of these solutions, to get a better feel for the behavior of the distance function. We will let δ_{MODEL} be the fraction of model points within ε pixels of edge points, and δ_{IMAGE} be the fraction of edge points within ε pixels of model points.

δ_{MODEL} has the values 0.711, 0.682, and 0.533 for solutions *(d1)*, *(d2)*, and *(d3)*, respectively. Most of the points on the *(d1)* solution lie on or near edge points, whereas a smaller fraction of the *(d2)* and *(d3)* points lie near edge points. This is due to two factors. First, portions of the long horizontals in *(d2)*, and particularly in *(d3)*, have no edge support, lowering their $g(d)_{MODEL}$ scores. Second, since *(d2)* has more pixels than *(d1)*, and since *(d3)* has more pixels than *(d2)*, these solutions must be near increasing numbers of edge pixels to achieve the same fraction as smaller solutions like *(d1)*.

δ_{IMAGE} has the values 0.138, 0.168, and 0.221 for solutions *(d1)*, *(d2)*, and *(d3)*, respectively. First, observe that the values for this term are much smaller than those for the other term. This is explained by the number of edge points in the chamfer image which do not correspond to the building, and will never be near a primitive solution, unless the primitive solution is particularly poor. Second, the edge points match more and more of the model points as we step through Figures 4-31, 4-32, and 4-33. In a loose sense, the two terms of Δ mediate one another, since the high δ_{IMAGE} scores associated with spurious models with large extents will be tied to low δ_{MODEL} scores associated with their lack of edge support.

Applying this distance metric to each of the 6 2-corners in Figure 4-26, the solutions *(b2)*, *(c3)*, *(d2)*, and *(f1)* were chosen, as well as *(a)* and *(e)* which were unique solutions and hence did not undergo the selection process. This set of selected solutions is shown in Figure 4-34. It should be stressed once more that these solutions are still represented solely as 2D points in image space with the appropriate topology; the conversion to 3D building models in object space is described in Chapter 5.

This running example illustrates the power of geometric constraints and relative size constraints in constructing increasingly detailed features for hypothesis generation. Beginning with a single line segment in Figure 4-5, the application of constraints to the feature generation process led to the creation of the set of 2D wireframes in Figure 4-34. For the entire Fort Hood scene depicted in Figure 4-1, 1315 2D wireframes were generated from the initial 1356 line segments, slightly less than one wireframe per segment on average. While the correct number of models is only a fraction of this number, the search space has been reduced to a degree amenable to more intensive verification methods, while allowing a wide range of viewing angles and different primitive shapes.

4.5. A summary of hypothesis generation

In some aspects, the selection process described in the previous section could have easily fallen into the category of "verification," a major topic of Chapter 5, which deals with choosing the best models of a scene from a set of hypotheses. Conversely, the primitive combination and extension algorithms in that chapter could be properly grouped under the "generation" heading, where "generation" refers to the creation of the set of hypotheses to be subjected to verification. Nonetheless, chamfer-based selection was described in this chapter, in order to maintain continuity in the description of the PIVOT process. In PIVOT, the boundary between hypothesis generation and verification is blurry; as we approach the goal of 3D object space volume generation, generation and verification phases alternate.

It can be argued that this lack of distinction between generation and verification at higher levels of the object detection and delineation problem is a consequence of the failure to adhere strictly to Principle 4. At several points in this chapter, algorithms have been presented which are effective in practice at reducing the combinatorics of feature generation, but use geometric tests which can discard potentially legal features. While the vanishing point geometry tests do not violate Principle 4, the relative distance constraints in Sections 4.2 and 4.3, the heuristics for selecting one edge for each placeholder edge slot in Section 4.4.2, and the Δ distance metric tests described in this section all violate the principle to varying degrees, since they can potentially discard a legal feature or primitive solution. If it were possible to generate geometrically legal features and primitive solutions without incurring combinatorial difficulties, there would be no need for verification tests during the generation process.

It should be noted, however, that the relative size constraints employed in this chapter represent a significant improvement over their predecessors in the literature; they do not require the specification of arbitrary threshold values or parameters. This is essential if an object detection and delineation system is intended to be fully automated, as PIVOT is. By using relative distances for accepting or rejecting potential corners, longer line segments are implicitly given more credence as possible object boundaries. This is in keeping with the notion of *non-accidentalness* elucidated by Witkin and Tenenbaum [108]; longer line segments are less likely to be the result of accidental alignments of intensity transitions in the image, and more likely to correspond to real structure in the image. From this viewpoint, it is natural to spend more processing effort (a larger window for corner search) to explain the presence of the line segment (find a geometrically legal corner which contains the line segment). While the relative size constraints by no means represent a formal probabilistic expression of the non-accidentalness of the intermediate features they constrain, they do provide an effective first-order approximation.

The notion of non-accidentalness for intermediate feature generation has only been partially explored in this chapter. Trihedral vertices, despite their viewpoint sensitivity mentioned in Section 4.3, represent another intermediate representation for which the non-accidentalness argument can be applied. The likelihood of three line segments in close proximity intersecting at a point is small, and so it can be argued that a trihedral vertex is unlikely to be the result of an accidental alignment of edges, and deserves more processing attention.

Another avenue of research involves *edge fragmentation*, the breaking of a single object boundary in the scene into multiple edges in the image. The difficulty here lies in developing a notion of non-accidentalness which does not violate Principle 4, as traditional perceptual grouping methods often do, while maintaining acceptable combinatorics. Another informal way of expressing this problem is this: while the probability of the accidental alignment of two edges is small, which "non-accident" caused it? Two aligned edges in close proximity might belong to the same object boundary; on the other hand, they might belong to *separate* objects in close proximity. Then again, they might belong to an object and a surface feature; roads and buildings are often adjacent, for example. As we saw in Section 2.3, local analysis is insufficient to discriminate these cases, yet analysis of all possible interpretations of the alignment is often combinatorially implausible. Mediating the non-accidentalness criteria with the combinatorially explosive nature of the search space remains an open question for low-level edge fragmentation analysis.

A key aspect of the intermediate feature generation methods described in this chapter is their heavy reliance on vanishing point information, not only for geometric constraints on features, but also for the inference of 3D structure. The work in this thesis uses vanishing points derived from the use of a photogrammetrically rigorous camera model, but recent work has also used vanishing points to reconstruct the 3D shape of labeled line drawings up to a scale factor [82]. In either case, vanishing points are the mechanism by which 3D information is computed from monocular views.

While many research issues remain open, the approach described in this chapter solves several important aspects of intermediate feature generation for monocular object detection and delineation algorithms. First, viewpoint independence under perspective geometry is achieved by using vanishing point analysis to constrain the search space. Second, the use of corners and 2-corners as intermediate features leads to the ability to model primitive shape by representing planar convex faces of the primitives. Third, a qualitative vanishing label analysis gives the ability to map the partial geometric descriptions provided by 2-corners into complete instances of primitive models. The resulting wireframe primitive descriptions are well suited to serve as the starting point for more sophisticated analysis, including the jump from 2D image space to 3D object space representations. These topics are addressed in detail in Chapter 5.

Chapter 5
Primitive Combination, Extension, and Verification

As with many other building extraction systems, PIVOT employs a data-driven hypothesize-and-test paradigm. In this processing strategy, early stages of image analysis create high-level object hypotheses from low-level image features, which are then subjected to verification procedures to select the best subset of hypotheses for a scene model. In Chapters 3 and 4, vanishing point detection and the resulting constraints for intermediate feature generation led to the creation of a set of geometrically consistent 2D wireframe primitives.

However, many of these primitives were produced by accidental alignments of edge fragments, and have no basis in the scene; many others result from multiple interpretations of the same group of edge fragments. Conversely, some objects in the scene are incompletely modeled by primitives; from near-nadir viewing angles, the rectangular portion of a peaked roof building is often invisible. As a consequence, further analysis of the initial set of primitive hypotheses is necessary before verification can proceed. This chapter focuses on three phases in the final mapping from an initial set of primitives to a 3D object space scene model: combination, extension, and verification.

One of the key ideas behind the use of primitives for 3D object modeling is that they represent a natural vocabulary for expressing complex shapes, eliminating the need for exhaustive representation of each unique shape that occurs in a scene. Primitive *combination* is the process by which primitive instances are joined together to create more complex objects. In this chapter, two types of combination are explored, one involving `peak-peak` combinations, the other involving `peak-rect` combinations.

The existence of a primitive instance can represent a strong cue for the presence of other primitives, if knowledge of certain types of primitive combinations is available. Primitive *extension* takes advantage of this idea, by using primitive instances as starting points for specialized searches for other primitives. In this chapter, one type of extension is explored; vertical extrusion, by which `peak` primitives are used as starting points for searches for supporting `rect` primitives, using the presence of vertical lines and shadow information for supporting evidence.

Finally, the topic of *verification* is explored. In contrast to the primitive generation process, which was driven by the imaging geometry, the verification of object models relies on the image photometry. Shadow analysis has been a popular verification technique in aerial image analysis, but it has typically been employed in image space. This chapter presents a new shadow verification method based in object space, as well as a new approach for qualitative evaluation of the illumination consistency of adjacent surfaces. These verification techniques are used in PIVOT's final processing phase to produce the final result: a 3D object space scene model in geographic coordinates.

5.1. Combining primitives in image space

Section 2.2 presented the motivation for Principle 2, which argued for the use of primitives for generic object modeling. To recapitulate, a key argument in favor of primitives was the ability to model the complexity of the visual world by composition of a few basic volumetric forms. For the detection and delineation of buildings in aerial images, it was argued in Sections 3.1 and 3.2 that rectangular and triangular volumes (abbreviated as `rect` and `peak`, respectively) provided sufficient representational power to model a significant majority of suburban and urban scenes.

One of the major drawbacks ascribed to the primitive modeling approach was the lack of robust techniques for extracting instances of primitives. That drawback was eliminated in Chapters 3 and 4 by the use of constraints derived from vanishing points, which resulted in the automatic generation of geometrically plausible primitive instances from a single image. By using vanishing point geometry derived from a rigorous camera model, no artificial limitations were placed on the primitive generation capability.

The other difficulty with the primitive modeling approach is the lack of methods for combining primitives in unconstrained environments. Current research on primitive-based modeling typically focuses on issues of recognition, rather than issues of detection and delineation. A common recognition problem assumes that a single multi-part object in a noise-free environment has been segmented into a line drawing, with the goal of inference of the semantic properties of the object from its constituent parts [96, 5]. In this situation, since the parts have been identified and since the segmentation is noise-free, detecting connections is straightforward. In uncontrolled scenes, however, segmentations are typically noisy, and it is often impossible to obtain a complete segmentation of an object part. Even reasonably accurate part segmentations may not necessarily form connected regions in the image.

PIVOT addresses the primitive combination problem by searching for primitives which have similarly shaped faces in close proximity. In particular, PIVOT's combination algorithm looks for two types of potential combinations; attachments of two `peak` primitives with aligned triangular faces, and the attachment of the "floor" rectangle of a `peak` primitive to the "roof" rectangle of a `rect` primitive. The first type of attachment is an example of handling fragmentation; the second type is an example of constructing new shapes from existing primitives.

In both cases, the algorithm operates on the 2D primitive wireframes, using both visible and invisible points. For each `peak` primitive, a range search is performed on the corner points of the triangular faces of the `peak` to find nearby points. If a match is found between all three points of one triangle and all three points of another, and the matches are oriented properly (i.e., the skyward point of one triangle is matched to the skyward point of the other triangle), then an attachment is formed between the two contributing `peak` primitives.

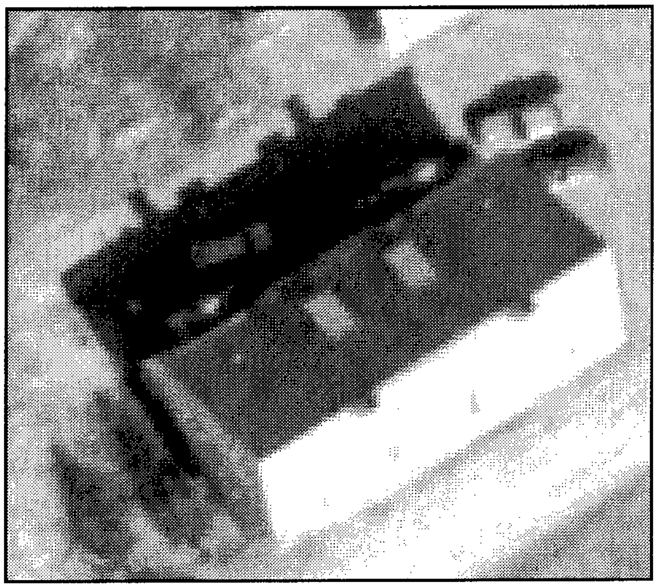


Figure 5-1: A building from FLAT_L

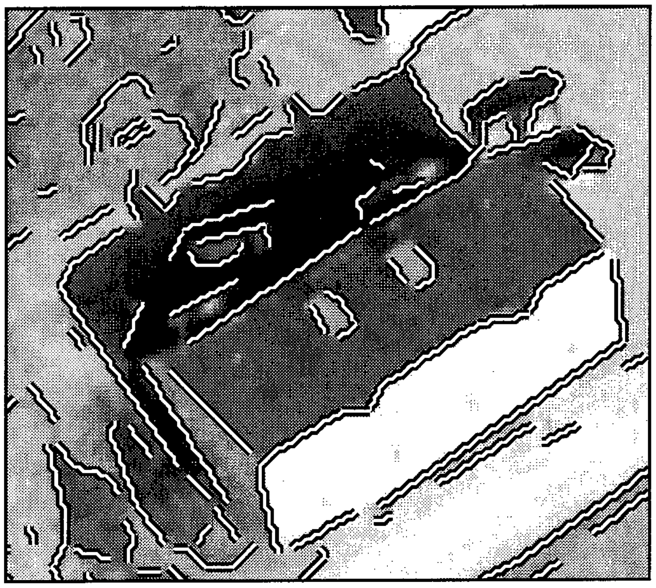


Figure 5-2: Results of edge extraction

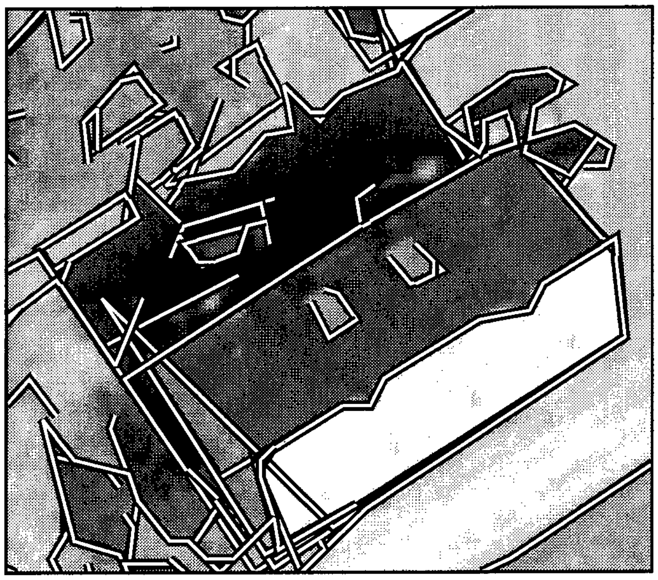


Figure 5-3: Generated corners

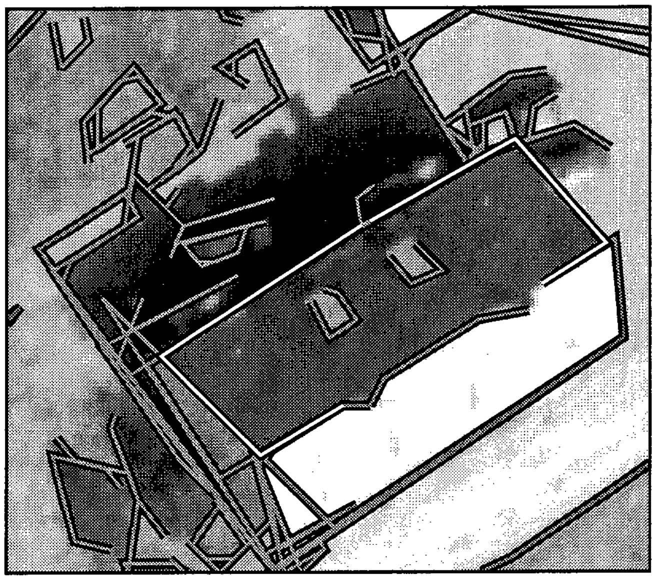


Figure 5-4: Generated 2-corners

Similarly, range searches are performed on the four corner points of the base of each `peak`; if a match can be found with all four roof points of a `rect`, then an attachment is formed.

Figure 5-1 shows a close-up of one building from the FLAT_L image in Figure 3-5. This peaked roof building will serve as a running example, to illustrate primitive combination and extension. Figure 5-2 shows the results of edge extraction on this image, and Figures 5-3 and 5-4 show the corners and 2-corners, respectively. Two 2-corners have been highlighted in Figure 5-4; the primitive generation for these 2-corners will be traced through the combination and extension process.

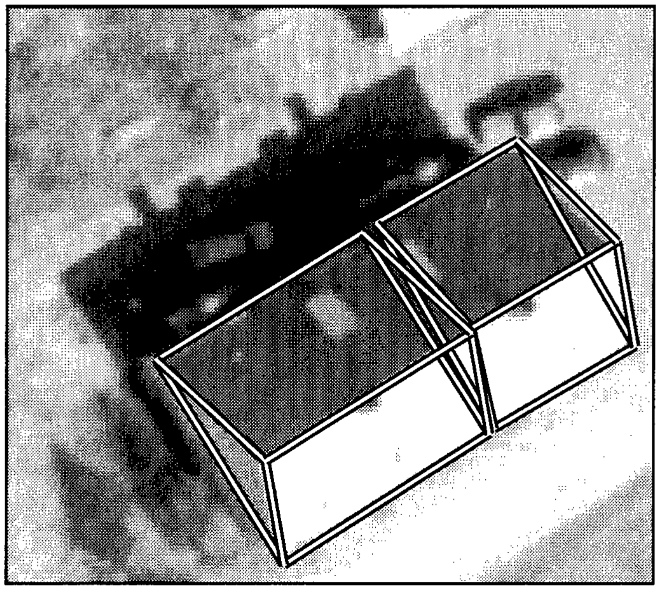


Figure 5-5: Two adjacent primitives

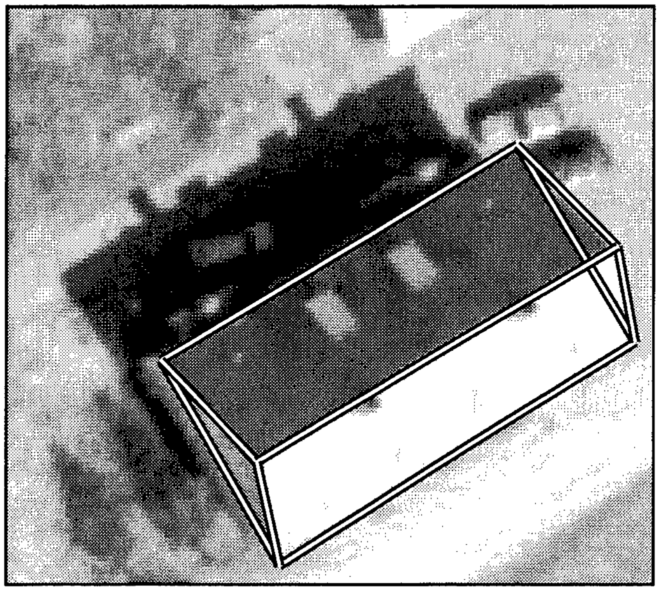


Figure 5-6: Result of primitive attachment

Figure 5-5 shows the `peak` primitives generated for each of the highlighted 2-corners in Figure 5-4. Since the interior triangular facets of each primitive are in close proximity at each vertex, PIVOT generates a third hypothesis by removing the the interior points and connecting the opposite triangular facets at each vertex. The result of this process is shown in Figure 5-6.

It should be noted that the current implementation of attachment in PIVOT is limited, primarily for combinatorial reasons. Only two adjacent `peak` primitives are allowed to be connected; long chains of more than two triangular prisms are prohibited. Further, since both `peak-peak` and `peak-rect` require each vertex of an aligned face to be matched, primitives whose faces do not align are not attached. Determining arbitrary connections between primitives is still an open problem, and is closely linked to the verification problem. Consider a set of primitive hypotheses, progressively smaller in size, which appear to be stacked like a set of children's toy blocks. To properly verify such a configuration, it would be necessary to test each possible combination of blocks for image support, which becomes computationally prohibitive as the number of blocks increases.

Nonetheless, it should be stressed that the ability to perform any primitive combination represents a significant advance, as it allows an object detection and delineation system to recover from low-level fragmentation errors in feature detection, as well as allowing the construction of more complex object shapes. It is also important to note that the primitive attachment is not *destructive*; i.e., it does not eliminate the original primitives from consideration during the hypothesis verification phase. This allows accidental primitive alignments to be rejected during verification if image analysis supports the interpretation of the primitives as individual entities rather than a combined one. Pointers are maintained from the new combined hypothesis back to its "parent" primitives, and vice versa. These pointers represent the *lineage* of the primitive; the use of lineage for verification is discussed in the next section.

5.2. Exploiting hypothesis lineage

The bulk of PIVOT's processing during early and intermediate feature generation is *noncompetitive*; every possible combination of low-level features is explored separately, without comparison against alternate interpretations of those same low-level features. Noncompetitive analysis is generally preferable, when constraints are available which place no limitations on the scene, image, or scene model. However, as we saw in Chapter 4, the vanishing point constraints did not completely eliminate multiple interpretations of the same underlying intermediate features. The first instance of *competitive* feature analysis occurred in Section 4.4.3, where a chamfer-based distance metric was used to select among a set of hypotheses to choose the one which best fit the edge data. In that case, each hypothesis competed against other hypotheses which shared the same 2-corner. The key idea was that each 2-corner corresponded to just one planar face of a primitive, and as a consequence, at most one of the hypotheses could be correct.

This is an example of a general competitive analysis method which PIVOT applies in situations where multiple geometrically legal models have been generated for a scene feature which, by definition, is unique. Because the scene feature is assumed to be unique, it follows that at most one model can properly represent that feature, and the rest can be discarded, saving processing effort for the best models for each feature. Determining which model best represents a feature involves rank ordering each model according to specific verification criteria. In the primitive selection example, the criterion was proximity of primitive edge points to image edge points. Of course, it may be the case that none of the generated models properly represent the scene feature at all, but the goal of competitive feature analysis is to choose the best model from a set of potential models, independent of whether the best model is good in absolute terms.

A useful technique in implementing competitive analysis in a data-driven object detection system is to maintain the *lineage* of features, or more specifically, to maintain pointers between each level of the hierarchy of intermediate features. For example, a 2-corner has pointers to the two corners which comprise it; conversely, each corner has a pointer to every 2-corner of which it is a component. By maintaining this lineage information, it becomes easy to determine sets of features which are in mutual competition.

The idea of competition between intermediate feature representations has been partially explored in other research. The most significant recent example for aerial image analysis comes from work on perceptual grouping by Mohan and Nevatia [74]. In this work, they defined features as *supportive* or *competitive* according to whether the features were linked by *part-of* relationships or shared component elements, respectively. A constraint satisfaction network based on Hopfield networks was then used to select the best mutually supportive set of features, treating the selection as an optimization problem. The interconnection weights for this network were chosen manually.

The difference between this kind of heuristic approach to competitive analysis and the approach taken by PIVOT lies in the notion of hypothesis uniqueness. Rather than employ arbitrary definitions of competitive relationships among intermediate features, PIVOT exploits the fact that certain mappings from feature to feature, and feature to object, must be unique. As described earlier, each 2-corner must map to at most one primitive; this is not an ad hoc choice, but a consequence of the geometry of primitives and 2-corners.

After application of the primitive combination process described in Section 5.1, PIVOT exploits the lineage of features again. Since an edge is assumed to correspond to a discontinuity between two surfaces in object space, there must be at most one primitive instance which correctly models the object or object component containing the two planar surfaces forming the edge. Hence, there can be at most one primitive corresponding to each edge. The lineage information could be used to trace back through primitives to 2-corners to corners to edges, to find the entire set of primitives associated with a particular edge *e*, but PIVOT uses a simpler and more robust approach. Since each 2-corner has been uniquely associated with a primitive, PIVOT uses the lineage pointers to find all 2-corners with central edge *e*; then, each of these 2-corners is mapped to its primitive, resulting in the set of competing primitives for edge *e*. This approach is more robust since it ensures that the same primitive is not chosen for two unrelated edges.

PIVOT selects the best primitive for each edge by scoring each of the competing primitives with an evaluation function and choosing the primitive with the best score. This evaluation function, which scores 2D image space primitives, is defined to be:

$$f(prim_{2D}) = w_E E(prim_{2D}) + w_H H(prim_{2D}) + w_\Delta \Delta(prim_{2D}) \tag{5.1}$$

w_E, w_H, and w_Δ are simply weights for each term of the evaluation function, and are each set to 1.0 in PIVOT. Each term of the function ranges from 0.0 to 1.0, so f ranges from 0.0 to 3.0. Δ is the distance function defined in Section 4.4.3 for evaluating the mutual fit of edge points and primitive points. The other two terms of f are new, and merit explanation.

While Δ measures the closeness of primitive points to edge points, E measures the actual image gradient support for the primitive wireframe. The first step in computing E involves hidden-line removal on the primitive, since we wish to evaluate the gradient support for only those edges which are expected to be visible. This computation can be implemented very efficiently for the 2D wireframe `peak` and `rect` primitives. Since both primitives are convex polyhedra, the direction of rotation around each face's boundary in both object space and image space must be the same for the face to be visible. These rotation tests can be easily implemented with Equation (A.24).

For each visible primitive edge, the edge gradient evaluation function used in the BUILD [68] and VHBUILD [67] systems is used to score the strength of the image gradient for that edge. This function performs a regression on the gradient maxima inside a window running along the edge, and computes the magnitude of the gradient transition across the regressed line. E is then defined to be the average of these scores, weighted by edge length.

H measures the homogeneity of the intensity values for each visible surface. Given the visible edges produced during the computation of E, finding the visible surfaces is trivial. The median intensity value for each visible surface is computed by scan-conversion of each surface polygon, and then the percentage of pixels for each surface with a brightness value within ε of that surface's median intensity is computed. For PIVOT, ε is set to 10; PIVOT operates on images with 256 gray levels. *H* is then defined to be the average of these percentages, weighted by surface area.

Certainly, the evaluation function *f* is not intended to capture fine-grained distinctions of photometric quality of the 2D primitive wireframes, as it does not account for occlusions or changes in surface texture. Instead, it is meant to provide a coarse assessment of hypothesis quality. To the extent that *E* and Δ capture the degree of edge support, and *H* captures the degree of surface homogeneity, then *f* will be able to distinguish hypotheses with good image support from those with bad image support. To better understand the behavior of *f*, we return briefly to an example from Chapter 4.

Consider the six primitive hypotheses in Figure 5-7. Since the 2-corners which generated these primitives all share the same central edge, they are considered to be in competition, and the evaluation function is applied to each of these primitives to select the one with the strongest image support. Table 5-1 shows the value of *f* for each of the primitives, along with the values of the terms of *f*.

First, we note that *(a)*, *(b2)*, and *(d2)*, qualitatively good models of the underlying scene structure, all have higher scores than *(c3)*, *(e)*, and *(f1)*, which are poor models of the scene. While none of the object delineations of the first three primitives are perfect, any of the three are accurate enough to serve as starting points for further analysis. In this case, *(b2)* is selected, and the other five primitives are discarded, since *(b2)* has the highest value of *f*. In examination of the values of specific terms, note that while *(e)* and *(f1)* both have higher values of Δ than *(d2)*, due to alignments with edges on adjacent buildings in the image, the values for *H* are quite low for both of these primitives. This reflects the fact that the visible polygons for these primitives do not exhibit strong intensity coherence. In contrast, *(d2)* has the highest value of *H* of any of the six primitives.

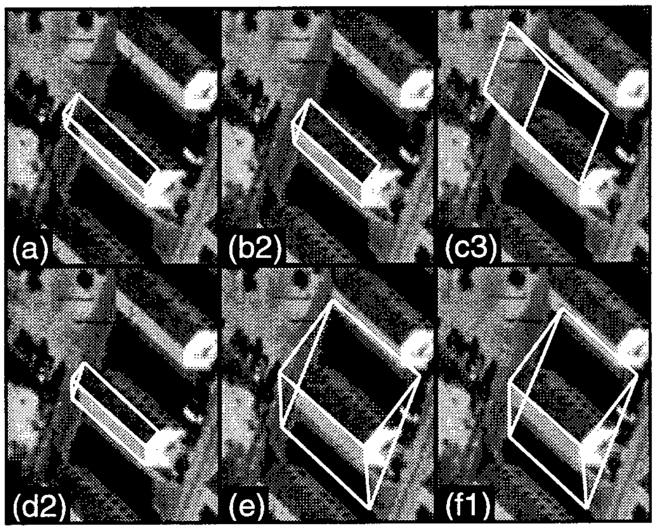


Figure 5-7: Competing primitives

2D primitive	*E*	*H*	Δ	*f*
(a)	0.718	0.654	0.541	1.913
(b2)	0.846	0.684	0.597	2.127
(c3)	0.292	0.280	0.387	0.959
(d2)	0.742	0.701	0.425	1.868
(e)	0.685	0.216	0.465	1.366
(f1)	0.728	0.200	0.493	1.421

Table 5-1: Evaluation scores for competing hypotheses

The use of qualitative evaluation functions in conjunction with lineage information provides a useful tool for pruning hypothesis space. In addition to its use in uniquely mapping 2-corners to primitives, and in this section for uniquely mapping edges to primitives, lineage information is exploited again in PIVOT's final verification phase. In that process, the lineage between the initial set of primitives and new primitives formed by combination and extension is used in conjunction with more intensive image analysis to produce PIVOT's final scene model. This process is described in detail in Section 5.5.2.

5.3. From image space primitives to object space models

Up to the present phase in the PIVOT processing chain, primitive representations have been instantiated as 2D image space wireframe models. This is largely due to the 2D nature of the vanishing point geometry used to create them, since the shapes of the primitives were defined by intersections of vanishing lines. It is also the case that intensity-based verification techniques typically require only lines and polygons defined in 2D coordinates, with no recourse to 3D information. The evaluation function in Section 5.2 is an example of such a verification technique; it makes no use of 3D information during its analysis.

While such techniques can be effective in early phases of object detection and delineation, where intermediate feature generation requires simple and effective analysis to constraint the search space, these methods often lack the ability to model effects which are essentially 3D in nature. Shadows, object-shadow occlusions, and object-object occlusions all represent powerful visual cues for the presence of scene structure, and all of these effects are inherently three-dimensional. Many prior approaches to shadow analysis in the aerial domain have assumed a nadir acquisition geometry, performing shadow mensuration in image space coordinates.

In Chapter 2, Principle 5 argued that these effects must be modeled in 3D to take full advantage of the structural information they provide for object detection. Another argument for producing a 3D object space scene model is that it is an enabling representation, particularly in the domain of aerial image analysis. By creating 3D models which are referenced to a geographic coordinate system, these models can be integrated with a wide range of geographic data, to provide increasingly detailed scene models for applications ranging from municipal planning to distributed simulation [60, 86].

To make the transition from 2D image space primitives to 3D object space models, a rigorous photogrammetric camera model is used in conjunction with digital elevation models (DEMs). The camera model is used to project image points to object space and vice versa, and the DEMs provide elevation data for the underlying terrain. The camera model is implemented as a toolkit providing several basic photogrammetric operations transparently over a variety of sensor platforms [65]. For the purposes of this work, the following basic photogrammetric operations are assumed to be defined for frame mapping photography:

- `image_to_world_DEM`: projects a 2D point in image coordinates to a 3D point in world coordinates, assuming that the point lies on the ground in the scene

- `image_to_world_elev`: projects a 2D point in image coordinates to a 3D point in world coordinates, assuming that the point lies at a specified elevation
- `image_vertical_height`: computes the object space height of a vertical line defined by two 2D points in image coordinates, assuming that the bottom point lies at a specified elevation
- `world_to_image`: projects a 3D point in world coordinates to a 2D point in image coordinates
- `view_vector`: computes the object space viewing vector through a 2D point in image coordinates

PIVOT generates six types of 3D models from 2D primitives. Two of these are direct translations of the `rect` and `peak` primitives to object space. Two others correspond to the `peak-peak` and `peak-rect` combinations described in Section 5.1. The remaining two correspond to incomplete `rect` and `peak` primitives; as described in Section 4.4.2, in some cases only one face of a primitive will be visible, due to the viewing angle or occlusion effects. The procedures for computing these six kinds of 3D models are discussed in turn.

1. The translation of a `rect` primitive to a 3D rectangular volume involves three steps. First, `image_to_world_DEM` is used to compute the 3D positions of the four ground points of the `rect`. Next, `image_vertical_height` is used to compute the object space heights of each vertical line in the wireframe. Finally, `image_to_world_elev` is used to compute the positions of the four roof points of the `rect`, using the vertical heights for elevation information.

2. The first step in the translation of a `peak` primitive is to use `image_to_world_DEM` to compute the 3D positions of the four ground points of the `peak`, just as in the `rect` case. However, computing the positions of the two peak points requires more work. Since it is assumed that the `peak` is symmetric with respect to a vertical plane along its horizontal axis, we begin by computing the center points of the bases of each triangular face. These object space points are then projected back to the image with `world_to_image`. `image_vertical_height` is then used with the apex of each triangle and its 2D base center point to obtain the heights of each triangle, and `image_to_world_elev` is used to compute the object space positions of both peak points, using the vertical height measurements.

3. A `peak-peak` attachment uses the same procedure as the basic `peak`. Since two `peak` primitives are combined by removing the two adjacent triangular faces and connecting corresponding vertices on opposite faces, the result is simply another `peak` primitive, which can be subjected to exactly the same procedure to translate it to object space.

4. The main difficulty in computing object space representations for `peak-rect` attachments lies in reconciling the ground points of the `peak` with the roof points of the `rect`, since these will rarely coincide perfectly in the image. PIVOT takes a simple approach; two models are generated, one which uses the `peak` ground points, the other which uses the `rect` roof points. These models will then be placed in competition in the final verification phase.

   In the first case, the 3D `peak` and `rect` models are computed as in (2) and (1), respectively. Then, the elevations of the roof points of the 3D `rect` are assigned to the corresponding ground points of the 3D `peak`, which has the effect of maintaining the (x,y)

position of each `peak` ground point while displacing it vertically. The ground points of the 3D `rect` are then assigned the (*x*,*y*) positions of the corresponding 3D `peak` ground points. Finally, the height displacements of the two peak points in the original 3D `peak` are used to displace the peak points with respect to the 3D `rect` roof.

In the second case, the 3D `rect` model is computed as in (1). Then, the 3D center points of the roof lines of the `rect` are computed, on the two roof lines which will form the bases of the triangular peak faces. `world_to_image` is then used to find the 2D image space positions of the center points, and `image_vertical_height` is used with these 2D center points and the 2D peak points to compute the height of this vertical line, assuming that the 2D center points lie at the elevation of the `rect` roof. This height displacement is then used to position the peak points above the `rect` roof.

5. For an incomplete `rect`, in which only the roof is visible, no height can be ascertained from object point measurements. So, `image_to_world_DEM` is used on the four corner points of the roof, and the ground points are assigned the same elevations, creating a rectangular volume in which the wireframe topology is maintained, but which has vertical edges of zero height. This volume is not allowed in the final scene model, but is instead marked as a candidate for shadow-based height mensuration, to be discussed in Section 5.4.

6. For an incomplete `peak` in which only one roof face is visible, there is still enough information to compute a complete 3D model. First, `image_to_world_DEM` is used on the two ground points of the roof face. Next, the vanishing point orientation vectors for the slanted lines of the face are obtained. `view_vector` is then used at each of the peak points of this roof to obtain a vector pointing from the camera through the object space position of the point. The intersection of the line through the viewing vector and the line through the ground point along the orientation vector gives the object space position of the peak point. The remaining two ground points can then be computed by using the symmetry of the `peak` primitive.

Due to noise in edge and corner extraction, and error in vanishing line intersections during the 2D primitive generation process, the 3D object space wireframes produced by the photogrammetric operations described here do not perfectly satisfy the expected building geometries; parallel faces and edges of buildings may not be perfectly parallel, and right-angled corners may be skewed. To correct the geometry, a least-squares fitting procedure is used to fit the 3D wireframes to idealized building models. The result of this process is a set of 3D building models, referenced to object space geographic coordinates, with lineage information pointing back to the original 2D wireframes. This information is used during the final verification phase, to be described in Section 5.5.2.

5.4. Primitive extension: extrusion methods

One of the principal motivations for using a primitive-based representation is its space efficiency. By modeling objects in the scene as collections of primitives, a large variety of structures can be modeled without the need for compilation of a correspondingly large model library. However, an equally important motivation for primitives, particularly in the context of monocular image analysis systems, is the leverage they provide to infer the presence of structure which is only partially visible or completely hidden from the camera.

This situation can arise in two different ways. In one case, partial support for a single primitive has been found in the image, but the viewing angle makes it impossible to determine the full extent of the primitive. For example, in an aerial photograph with a nadir or near-nadir acquisition geometry, only the roofs of flat rectangular buildings will be visible, making it impossible to ascertain the height of the building from its boundary alone. In fact, as mentioned in Section 5.3, PIVOT initially generates rectangular volumes with no vertical height in this case. In the other case, the extent of a primitive may be fully determined, but the viewing angle or occlusions prevent an attached primitive from being extracted. An example in this case is a peaked roof building in a nadir mapping photograph, where the supporting rectangular base of the building may not be visible. Of course, both cases are similar, in that there is an underlying expectation that more structure may be present in the scene, based on partial evidence, than that which can be derived from the image through object boundary analysis alone.

The general idea behind primitive *extension* is the completion of partial descriptions of primitives, or collections of primitives, by assuming the partial description is accurate and using it as a starting point to hypothesize the missing information. PIVOT currently performs two types of extensions, both of which consist of extrusions along vertical axes in object space. In the first extension technique, the zero-height rectangular volumes from the previous section and triangular volumes are extruded by shadow mensuration. In the second extension technique, peak volumes are extruded by vertical line analysis. While these represent a limited set of the possible types of extrusions that could be performed, they are sufficient for illustrating the benefits of primitive extension for generating complete object hypotheses.

Before beginning a detailed discussion of the primitive extrusion procedures, it is worth noting that some care must be taken in order to properly extrude primitives. Simply extruding primitives vertically in object space is not sufficient, since the original boundaries of the primitive will project to a different location in the image due to their change in elevation. To properly extrude a primitive vertically, it is necessary to first use `world_to_image` to project the original primitive to the image, and then apply `image_to_world_elev` with the extrusion elevation on the appropriate image points to compute an extruded volume with the proper height and placement in object space. This combined operation will be identified as `extrude_by_height` in subsequent sections. In both cases, the goal of the automated vertical extrusion process is to estimate the vertical height of the extrusion, which is then used with `extrude_by_height` to perform the actual extrusion.

5.4.1. Extrusion by vertical line analysis

One approach for measuring the height of an extrusion in object space involves the measurement of vertical lines. This basic idea was used in the VHBUILD system [67], which used an oriented edge-finding technique at the corners of rooftop polygons to find vertical lines. These vertical lines were then measured with `image_vertical_height` to determine the height of the building in object space. As in PIVOT, the vertical vanishing point was directly computed from the camera model.

While the specifics of PIVOT's extrusion method are different, the basic principle of finding vertical lines at roof corner points remains the same. For each `peak` primitive, PIVOT begins by locating the subset of the four roof corner points which could potentially have vertical edges visible from the camera. These points are then mapped to image space via `world_to_image`, and a range search is performed on each image space point, using a range query database consisting only of edges with vertical vanishing point labels. The combined result of these searches is a list of vertical edges and the corner points with which they are paired.

PIVOT then selects one of these vertical edges, based on the quality of the vertical's fit to the image, its angular deviation with respect to the vertical vanishing point, and its proximity to the corner point. PIVOT's vertical edge evaluation function is:

$$f(edge_v) = w_A A(edge_v) + w_B B(edge_v) + w_C C(edge_v) \tag{5.2}$$

w_A, w_B, and w_C are simply weights for each term of the evaluation function, and are each set to 1.0 in PIVOT. Each term of the function ranges from 0.0 to 1.0, so f ranges from 0.0 to 3.0. A is the edge gradient evaluation function used in BUILD and VHBUILD, and was previously described in Section 5.2.

B measures the angular deviation of an extended line of a vertical line segment with respect to the vertical vanishing point. The angular deviation θ is measured by computing the angle formed by the line segment with an imaginary line running between the midpoint of the line segment and the vertical vanishing point. Figure 5-8 depicts this computation. B is then defined to be $1 - (\theta / \theta_{MAX})$, where θ_{MAX} is the maximum allowable angular tolerance for a line segment to be labeled as vertical. In PIVOT, θ_{MAX} is 20 degrees.

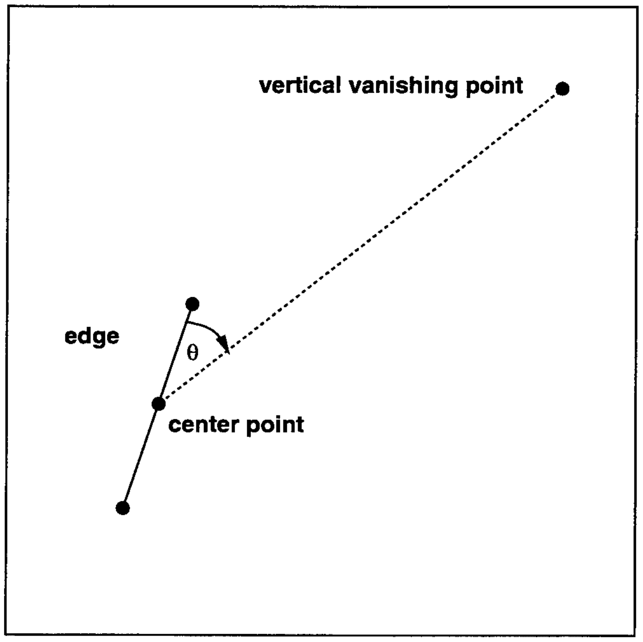


Figure 5-8: Angular deviation for a vertical edge

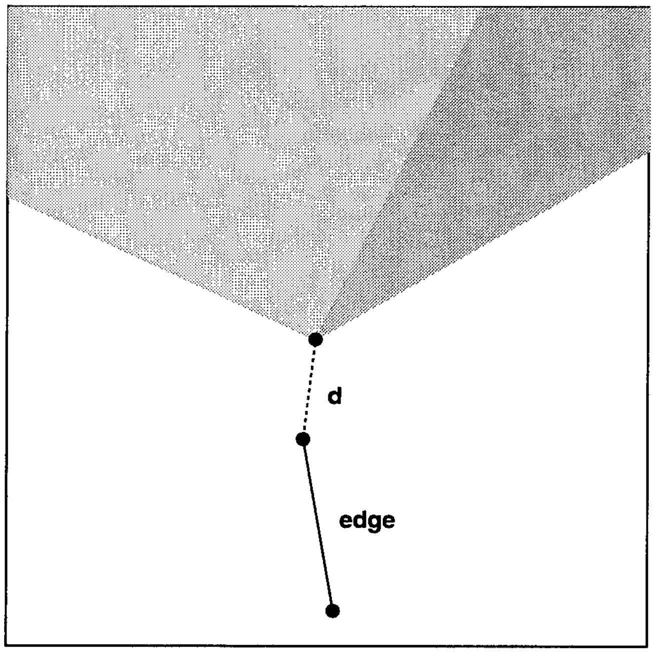


Figure 5-9: Proximity of a vertical to a corner point

C measures the proximity of a vertical edge to the corner point with which it is expected to coincide. First, the distance d from the corner point of the `peak` primitive to the nearest endpoint of the vertical edge is computed, as shown in Figure 5-9. Then, C is defined to be $(d_{MAX}-d) / d_{MAX}$, where d_{MAX} is the maximum radius of the range search window used to find the vertical edges. In PIVOT, the query rectangle for this vertical edge search is a fixed window of radius 8, and hence d_{MAX} is $8\sqrt{2}$.

Once f_v has been computed for each vertical edge, the edge with the highest value is selected, along with its associated `peak` corner point. `image_vertical_height` is used to measure the vertical distance in object space between the corner point and the endpoint of the vertical edge which is furthest away from the corner point. The resulting height is then supplied to `extrude_by_height` to extrude a rectangular support volume for the `peak` volume.

PIVOT uses this vertical-based extrusion process exclusively on `peak` primitives. There is no need to perform vertical-based extrusion on `rect` primitives, since for nadir views, verticals would not be visible, and for oblique views, visible verticals would already have been used in the 2D primitive generation process.

5.4.2. Extrusion by shadow mensuration

One of the classical tools for height determination in aerial photographs is shadow mensuration. When the angle of illumination is known, simple trigonometry can be applied to an object line and its cast shadow line to determine the height of the object line. This idea was first explored in building extraction work by Huertas and Nevatia [42] and Irvin and McKeown [46], to derive image-space based estimates of height for flat rectilinear structures. More recently, Lin and Nevatia [55] used shadows to compute the heights of flat rectilinear structures in object space.

This section presents an approach for shadow mensuration which also operates in object space, but which works on `peak` primitives as well as `rect` primitives. The approach can be summarized as an iterative extrusion of the primitive from the ground plane until the projected shadow boundary of this extruded primitive best coincides with an intensity discontinuity in the image, having the expected dark-to-light transition from shadow to ground. This extrusion process can be employed on both `rect` and `peak` primitives, including the incomplete `rect` primitives generated in Section 5.3. A detailed description of this extrusion process follows.

First, the solar illumination vector is computed from the acquisition parameters of the aerial photograph [94]. Next, all non-floor shadow casting edges are located for the primitive, since these are the edges which will produce a shadow on the ground. This computation is easily carried out for simple convex polyhedra by a counting process. Each edge of the polyhedron is initialized to zero. Then, for each polygon of the polyhedron which faces the light source, each edge of the polygon is incremented. Edges which have a value of 1 at the conclusion of this process are shadow casters, since only one of the two faces which created them are directly illuminated by the light source.

The next step involves determining a suitable height increment for the extrusion process. PIVOT computes an object space increment z such that the corresponding increment of the shadow boundary in image space is at most one pixel. This is done by computing the height of a vertical at one corner of the primitive, under the assumption that its shadow is one pixel long in image space.

Then, PIVOT iteratively generates extruded building models, starting with a building model of height z and incrementing by z every iteration. At each iteration, the projected shadow boundary is computed by an object space projection of the shadow-casting edges along the illumination vector to the ground plane (assumed to lie at the same elevation as the building base). If the shadow line produced by a shadow-casting edge is occluded by the building, it is discarded. This leaves only that portion of the shadow boundary which is unoccluded by the shadow-casting object.

For each iteration, the unoccluded shadow boundary is evaluated, by computing a length-weighted average of evaluation scores for each segment of the shadow boundary. Each segment is evaluated using the following function:

$$f(edge_S) = w_A(edge_S)\, A(edge_S) \qquad (5.3)$$

A is simply the edge gradient evaluation function described in Section 5.4.1. w_A is an edge-dependent weighting term which ranges linearly from -1 to 1, and accounts for the expected dark-to-light transition across the shadow boundary away from the building. w_A is computed by measuring the median intensity on both sides of the line segment and computing the difference, where a brightness increase of 10 or more intensity levels gives a score of 1, and a brightness decrease of 10 or more intensity levels gives a score of -1, ranging linearly between these two values for median brightness differences of less than 10 intensity levels. In summary, the magnitude of $f(edge_S)$ corresponds to edge strength; positive or negative sign corresponds to good or bad dark-to-light intensity transitions, respectively.

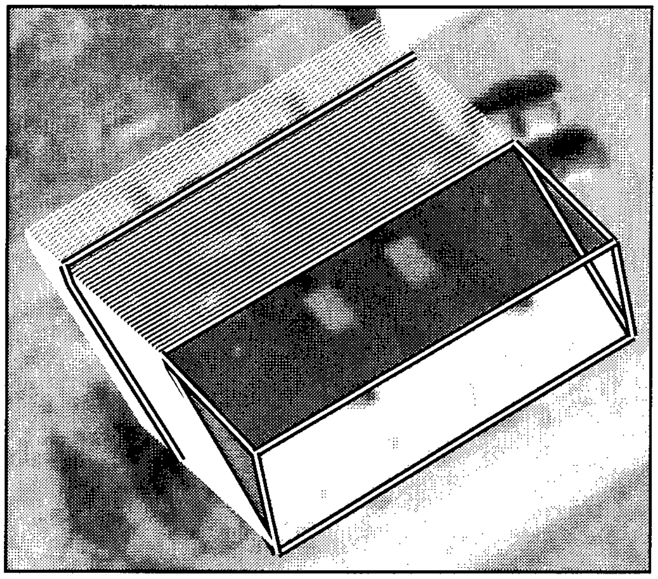


Figure 5-10: Search for shadow boundary

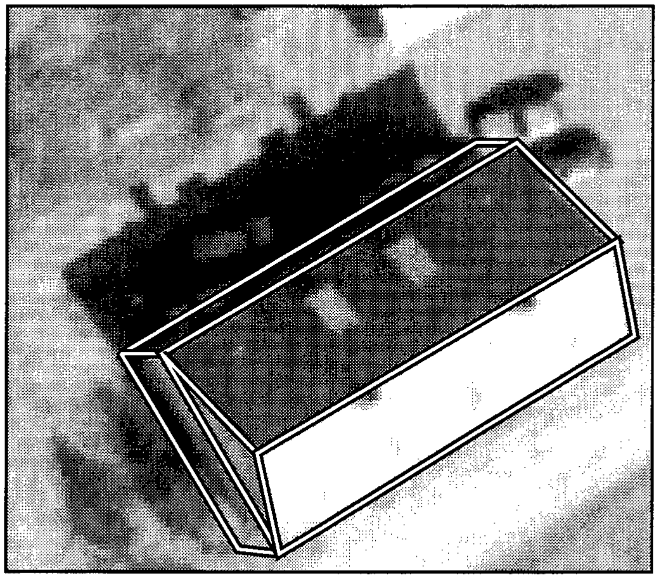


Figure 5-11: Result of shadow-based extrusion

The normalized length-weighted average of this evaluation function is computed over all unoccluded shadow edges in a shadow boundary; the shadow boundary with the highest score then selects the height for the extrusion, and `extrude_by_height` is used with this height to generate the extruded building. Figures 5-10 and 5-11 illustrate this shadow-based extrusion method for the running example in this chapter. Figure 5-10 shows the projected shadow boundaries generated during the extrusion process, with the highest scoring boundary highlighted. Figure 5-11 shows the result of using the height determined by the selected boundary to extrude the `peak` primitive to the ground.

It should be noted that this approach does not attempt to decide whether the shadow boundary with the highest score is good in absolute terms; every input hypothesis is extruded to form a new hypothesis, regardless of the quality of the shadow support each input hypothesis has. For example, a false positive primitive will still be extruded, and the shadow boundary with the highest score for this primitive will still be used to generate a new building volume, even though the score may be quite low since the false positive will likely have little or no image support for a shadow region. The task of deciding whether an extruded primitive and its projected shadow are good in absolute terms is left to a more comprehensive hypothesis verification technique, described in Section 5.5.

Currently, PIVOT performs a fixed number of iterations (40) for the shadow extrusion process. Alternatively, a maximum building height could be specified by a user on a case-by-case basis, but PIVOT is intended to be a fully-automated system. Another approach would be to let PIVOT iterate until the building extrusion or its shadow projection reached the image border, but this would use excessive computation time. A better approach would be to use automated techniques for determining termination points automatically, such as the sequence analysis methods used in BUILD and VHBUILD [110].

Nonetheless, the benefits of both shadow-based and vertical-based extrusion are clear. In situations where portions of buildings are partially or totally obscured, or where the camera viewing angle prohibits complete generation of object structure from the object boundary alone, primitive extension methods provide a powerful tool for creating new object hypotheses from other geometric cues.

Both the vertical-based and shadow-based extrusion methods maintain lineage pointers to the original building hypotheses; this information is used in PIVOT's verification process, to be described in the next section. At this stage of processing, PIVOT has completed all hypothesis generation steps, and the remainder of its processing is focused solely on verifying the subset of hypotheses which best constitutes an accurate scene model.

5.5. Hypothesis verification

As mentioned in Section 4.5, the existence of a hypothesis verification phase in an object detection and delineation system is a direct consequence of the inability to generate only the minimal set of hypotheses necessary to model the scene. In PIVOT, vanishing point constraints serve as a powerful mechanism for significantly pruning the search space for valid hypotheses; even so, many of the object space volumes PIVOT generates have no foundation in the scene, resulting from accidental alignments of natural features whose vanishing point labels satisfy the corner generation constraints.

Since all of PIVOT's remaining hypotheses are geometrically legal, other means must be employed to distinguish good building hypotheses from bad ones. Traditionally, monocular building detection systems have used photometric properties as the principal component of their verification algorithms. Shadow analysis has been the most popular monocular verification technique, since shadows often represent prominent features in an image, and even simple image processing methods are sufficient for locating shadows and/or extrapolating them from object hypotheses. PIVOT uses a *shadow-based* evaluation technique as one component of its hypothesis verification function.

However, shadows are not the only photometric effects which can be exploited for hypothesis evaluation. If the position of the light source is known, then the qualitative appearance of object hypotheses can be evaluated with respect to the light source. Specifically, for Lambertian objects illuminated by a point light source, we expect specific changes in the intensity of the reflected light as the object surface normals change angles with respect to the light source. PIVOT employs an *illumination-based* technique in its verification function as well, to ensure that changes in reflected light are consistent across polyhedral facets.

Intensity changes and shadows are inherently 3D illumination phenomena, and hence PIVOT models them in object space, in keeping with Principle 5. Other photometric effects, such as sharp changes in gradient at object boundaries and the smoothness of the intensity surface inside polygonal object faces, are naturally modeled in 2D. For these latter effects, PIVOT uses photometric measures which appeared earlier in Section 5.2; the edge gradient support function E, and the surface intensity homogeneity function H.

In the next subsection, detailed descriptions of the shadow-based and illumination-based evaluation components of PIVOT's verification function are presented. Following those descriptions, PIVOT's verification algorithm is presented in its entirety in Section 5.5.2.

5.5.1. Evaluating object photometry

Shadow analysis for hypothesis verification in monocular building extraction has been an active research topic in recent years. Huertas and Nevatia [42] used shadows for verifying rectangular image space hypotheses, as well as for intermediate feature analysis and height mensuration. Irvin and McKeown [46] also used, shadows for verifying rectangular image space hypotheses, as well as hypothesis generation and grouping. More recently, Lin and Nevatia [55] used shadows for verifying rectangular volume hypotheses, as well as height mensuration in object space.

These methods generally take one of two approaches towards the use of shadows for verification. In the first approach, the expected shadow-casting sides of each hypothesis is examined for the presence of a corresponding dark region, the existence of which is assumed to support the object hypothesis. This essentially intensity-based approach to shadow verification contrasts with the geometry-based approach, which looks for edge and corner features in the image which correspond to the expected shadow boundary of an object hypothesis.

PIVOT employs a method which combines aspects of both approaches to evaluate the shadow support for a building model. The method is intensity-based, in that it checks to see that the expected dark-to-light transition is present from the shadow region to the surrounding ground. The method is geometry-based, in that the shadow and ground regions are determined by projection of the 3D model.

PIVOT tests shadow-to-ground consistency by computing the median intensities inside two regions in the image, the *shadow region* and the *ground region*. The shadow region is created by projecting all of the building points along the direction of the solar illumination vector to the ground plane, and then computing the enclosing polygon of these ground points. This polygon is then projected back to image space, and the polygon enclosing the building boundary is subtracted from it, leaving the shadow region.

A fixed-width distance transform is then run on the shadow region, to find a polygon which encloses the shadow region with a fixed separation width (4 pixels in PIVOT). Both the building boundary region and the shadow region are subtracted from this polygon, leaving the ground region. If the median intensity of the ground region is less than that of the shadow region, the building hypothesis is rejected. Figure 5-12 shows the shadow region and ground region for a peaked roof building, as computed by projection of the building points and polygon subtraction.

If the median intensity of the ground region (I_{GRD}) is equal to or greater than that of the shadow region (I_{SDW}), then a *shadow consistency score* is computed:

$$S_{HYP} = \max\left(1.0, \frac{I_{GRD} - I_{SDW}}{I_S}\right) \tag{5.4}$$

where I_S is a scaling constant, set to 10 in PIVOT. With this setting, small differences in the median intensities of the shadow and ground regions lead to lower shadow consistency scores.

Unlike shadow analysis, illumination-based approaches have seen little use in hypothesis verification algorithms, and in the broader topic of feature extraction from aerial images. Bro-Nielsen's work [13] is a notable exception; he developed an elaborate solar illumination model which accounted for atmospheric scattering, breaking illumination into direct and indirect components. This model was then used to compute luminance features for polygons in a region-based classification system. This luminance information was not employed in a verification procedure, however.

Qualitatively, we expect that surfaces which face the sun should be brighter than those which face further away, assuming identical surface material compositions. PIVOT uses a simple computer graphics model for illumination, treating surfaces as Lambertian reflectors and the sun as a point light source, and breaking reflected light into ambient and diffuse components [25]:

$$I = I_a k_a + I_p k_d (\mathbf{n} \cdot \mathbf{l}) \tag{5.5}$$

I_a is the intensity of the ambient light, and k_a is a material-dependent coefficient which determines the percentage of ambient light which is reflected from a surface. I_p is the intensity of the point light source, and k_d is a material-dependent coefficient which determines the percentage of direct light which is reflected. **n** is the surface normal, and **l** is the solar illumination vector, determined from the image acquisition parameters.

For each surface of a building model which is visible from the camera, I (the median surface intensity) and $(\mathbf{n}\cdot\mathbf{l})$ are known; the terms $I_a k_a$ and $I_p k_d$ are unknown. But, for two or more surfaces, a least-squares line fit solves for the unknown terms. Then, if $I_p k_d \leq 0$, the building hypothesis is rejected, since its surfaces appear brighter as they face further away from the sun.

PIVOT tests the illumination consistency of roof and wall surfaces separately, since building roofs are frequently composed of a different material than walls. This approach is a simple and effective way to quickly prune large numbers of hypotheses which may be formed by accidental alignments of image features. For example, shadow regions often have vanishing point geometries which are consistent with respect to the primitive models. The primitives they form, however, do not have consistent illumination, since the shadow region is typically included as a wall or roof polygon in the hypothesized building model, and this polygon is too dark with respect to the adjacent surfaces of the model.

Figure 5-13 illustrates the comparison of surface normals with the solar illumination vector, where $\cos\theta = (\mathbf{n}\cdot\mathbf{l})$. Note that only three of the four visible surfaces form angles of less than 90 degrees with the solar illumination vector; the fourth surface would be included in a plane fit with $(\mathbf{n}\cdot\mathbf{l})$ set to 0, since it reflects only ambient light.

If the roof and wall illumination fits are consistent, then an *illumination consistency score* is computed for the building hypothesis:

$$I_{HYP} = \begin{cases} 0.5 & \text{if } n<2 \\ \max(0, \operatorname{sgn}(I_p k_d)) & \text{otherwise} \end{cases} \tag{5.6}$$

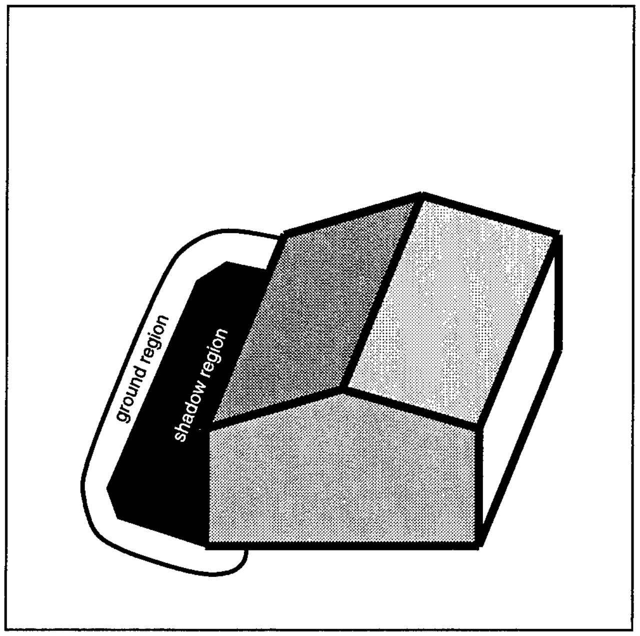


Figure 5-12: Evaluating shadow-to-ground consistency

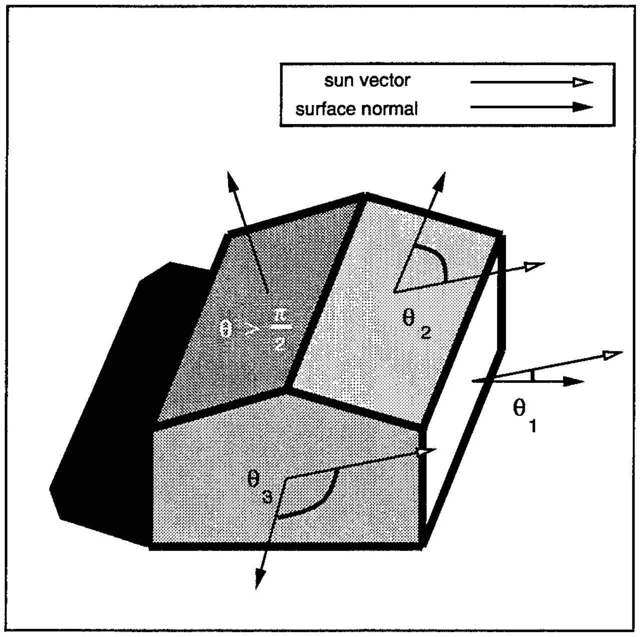


Figure 5-13: Evaluating inter-surface intensities

where n is the number of visible surfaces for the building hypothesis, and $I_p k_d$ is the result of a least-squares fit to both wall and roof surfaces simultaneously. sgn is the standard sign function, returning 0 if its argument is 0, -1 if its argument is negative, and 1 if its argument is positive. It follows that I_{HYP} takes on one of three possible values for any building hypothesis: 0 if the illumination fit over both wall and roof surfaces appears to be inconsistent, 0.5 if only one building surface is visible, and 1 if the illumination fit over all surfaces appears to be consistent.

In both shadow-based and illumination-based evaluation, PIVOT rejects hypotheses which have inconsistent shadow support or intensity changes across surfaces. For the remaining hypotheses, S_{HYP} and I_{HYP} are computed as measures which reflect the confidence of each hypothesis based on its shadow support and intensity consistency, respectively.

5.5.2. A verification method

PIVOT's final verification phase, like the intermediate pruning phases encountered in earlier stages of the hypothesis generation process, is based on the use of lineage information to keep only the best hypothesis corresponding to a given set of image features. PIVOT constructs sets of building hypotheses which are related by lineage, and then selects the best representative from each set according to a confidence score based on several measures of hypothesis quality.

The confidence score for a building hypothesis is:

$$C_{HYP} = w_I I_{HYP} + w_S S_{HYP} + w_E E_{HYP} + w_H H_{HYP} + w_B B_{HYP} \tag{5.7}$$

The w_i in this expression are simply weighting coefficients, and are each set to 1.0 in PIVOT. Each term ranges from 0 to 1, so C_{HYP} ranges from 0, the weakest confidence score, to 5, the best possible confidence score. I_{HYP} and S_{HYP} are the illumination consistency and shadow consistency scoring functions, respectively, as defined in Section 5.5.1. E_{HYP} and H_{HYP} are the edge gradient evaluation function and the surface homogeneity scoring function, respectively, as defined in Section 5.2.

B_{HYP} is the *believability* function; it incorporates domain-dependent knowledge about object extent. Since PIVOT operates on aerial imagery covering urban and suburban areas, we can reasonably expect that its building hypotheses should correspond to actual buildings in the imaged scenes, and as such, they should possess at least certain minimal dimensions. B_{HYP} is defined in terms of minimal extents each building hypothesis is expected to have, and penalizes hypotheses which do not possess the minimal extents. Specifically:

$$\begin{aligned}
B_{HYP} &= \frac{1}{3}(B_{length} + B_{width} + B_{height}) \qquad (5.8)\\
B_{length} &= \min(1, \frac{length}{minl})\\
B_{width} &= \min(1, \frac{width}{minw})\\
B_{height} &= \min(1, \frac{height}{minh})
\end{aligned}$$

length, *width*, and *height* are the dimensions of the building hypothesis. In PIVOT, *minl* and *minw* are set to 5 meters, and *minh* is set to 2 meters; this means that building hypotheses with lengths or widths less than 5 meters, or heights less than 2 meters, will be penalized by the scoring function.

C_{HYP} is computed for every building hypothesis, and then the lineage information stored in each hypothesis is used to construct a graph of hypotheses and their dependencies on one another. Recall that building hypotheses can be generated by several mechanisms; in addition to the basic `peak` and `rect` primitives which form the atomic units for constructing building models, there can also exist hypotheses which are the result of combination of primitives, as well as those which result from extension of primitives. PIVOT uses this lineage information in an iterative algorithm which selects the best hypotheses based on confidence scores and competitive analysis. This algorithm begins by sorting all building hypotheses by decreasing confidence score, so that the highest-scoring hypothesis is at the beginning of the hypothesis list. PIVOT then iterates through the list, choosing the hypothesis with the highest score, and removing all competing hypotheses from the list. This process continues until the entire list has been traversed.

To illustrate the operation of this verification algorithm, Figure 5-14 graphically depicts a set of building hypotheses with their associated confidence scores, using pointers to indicate descendant lineage. In this example, primitives A through G constitute the "atomic" primitives. H, J, K, and L were generated by combination of primitives, while I, M, and N were generated by extrusion.

Figure 5-15 shows the result of the first iteration of the verification algorithm. Hypothesis E has the highest confidence score, so it is retained. Hypothesis K was constructed by attachment of E and F, so it is now discarded from the list since E had a higher confidence score by itself than as a component of K. Note that F is not discarded, since it is not directly competing with E. Figure 5-16 shows the result of the second iteration of the verification algorithm. The next highest score is associated with hypothesis M, so it is retained. M was constructed by extrusion of J, which itself was comprised of C and D, so these three hypothesis are discarded from the list, as the combined interpretation represented by M has a higher score than any of the three components. Hypothesis I was constructed by extrusion of C; but since C has been eliminated, I is eliminated as well.

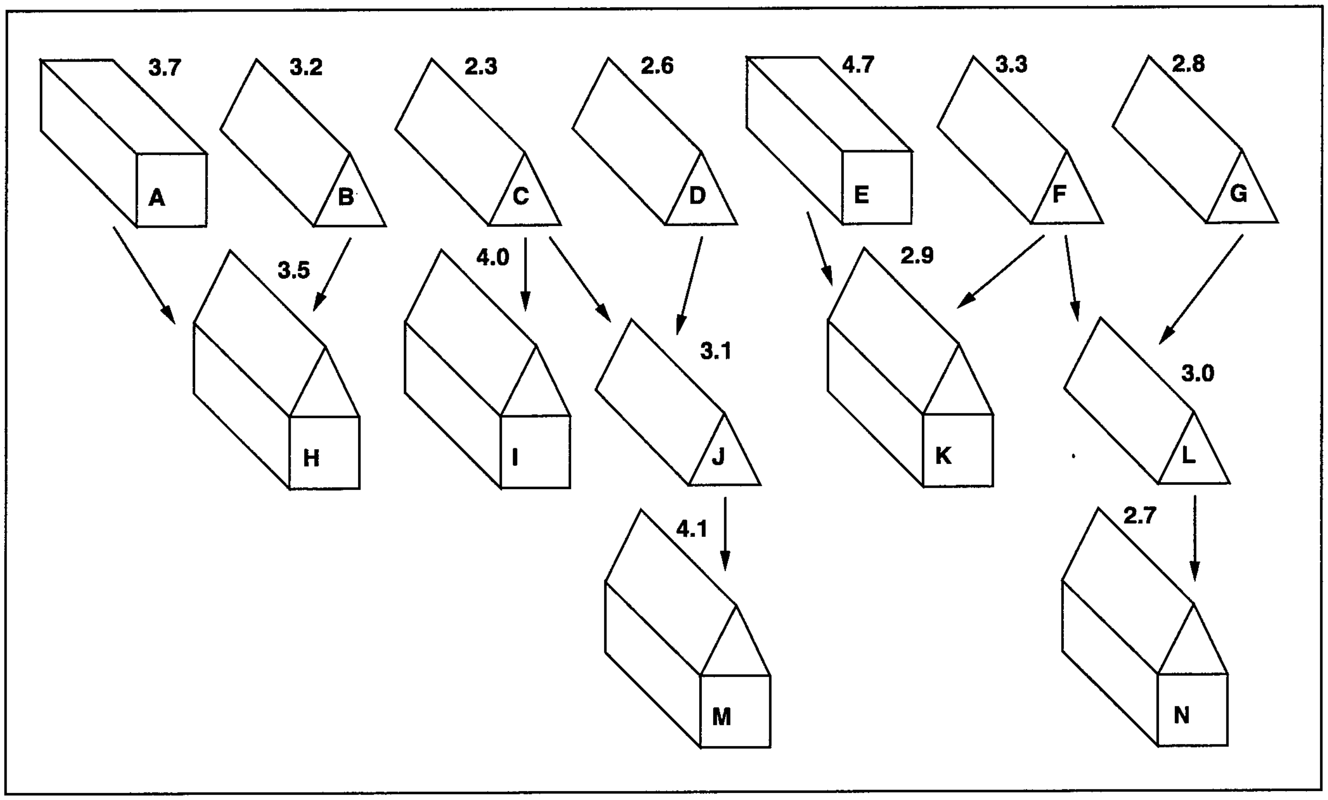


Figure 5-14: A set of building hypotheses with confidence scores and lineage

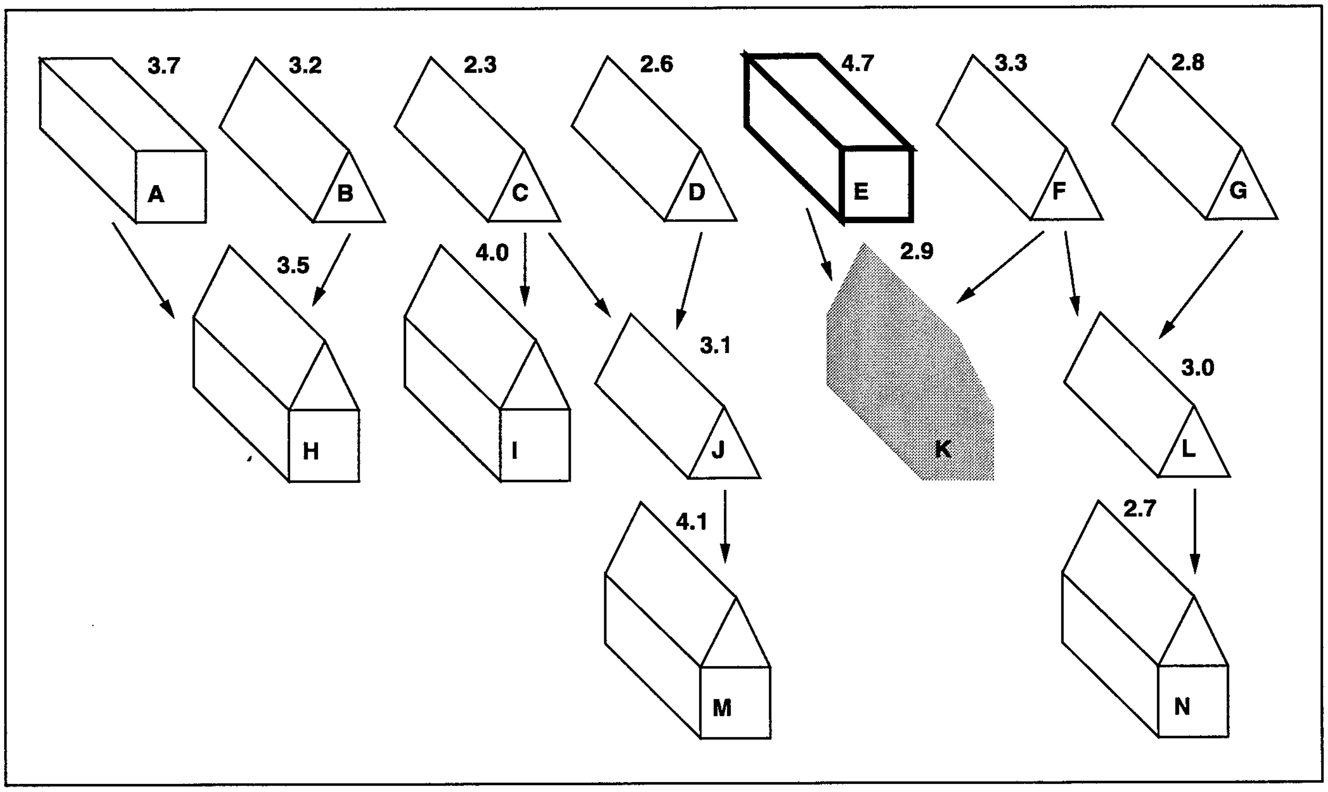


Figure 5-15: First iteration of lineage-based verification algorithm

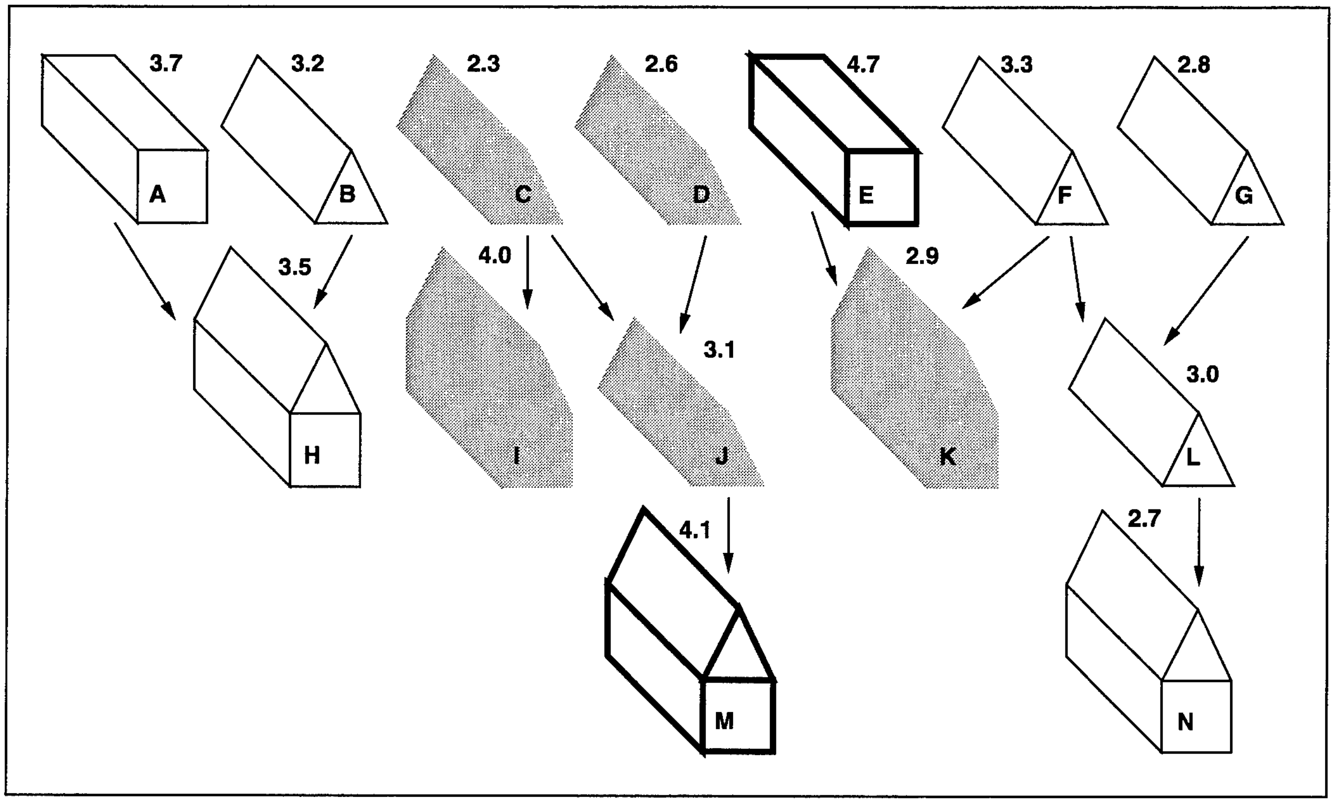


Figure 5-16: Second iteration of lineage-based verification algorithm

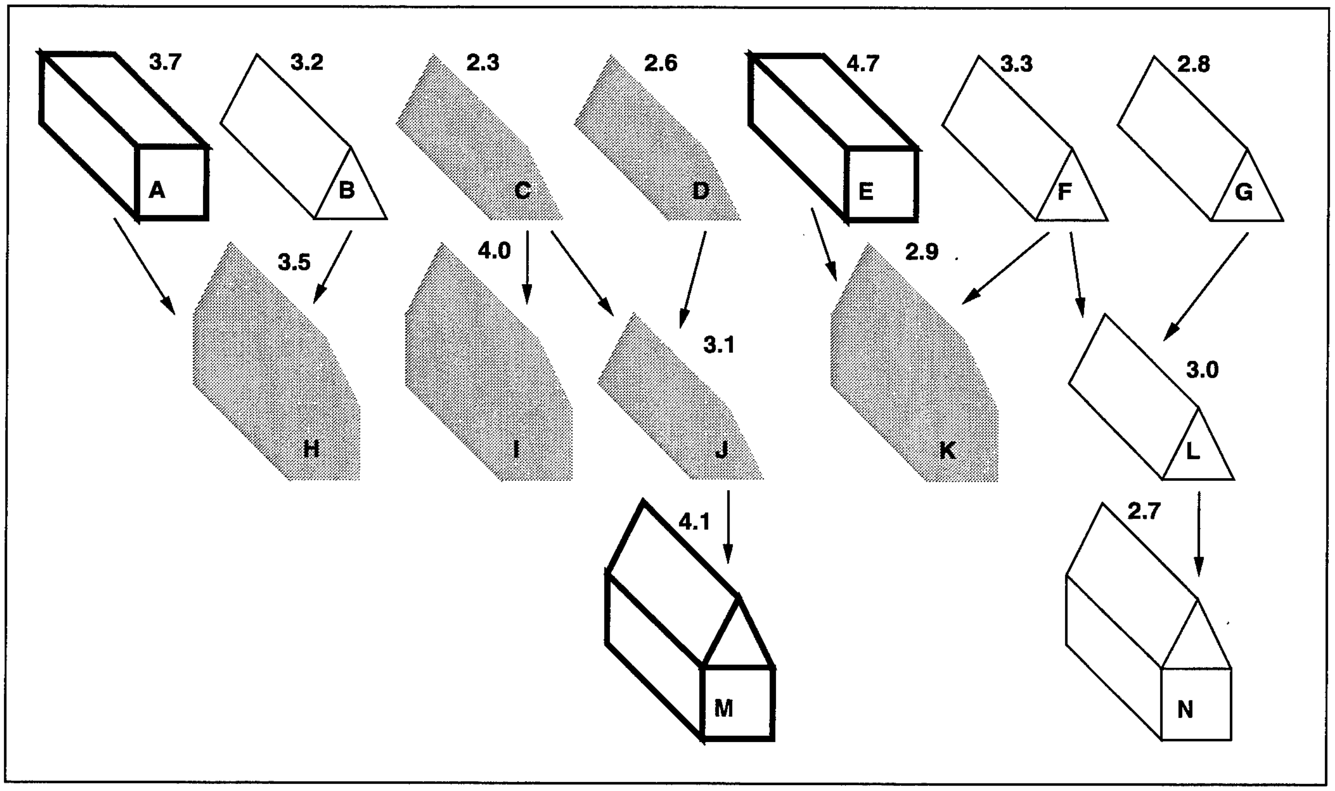


Figure 5-17: Third iteration of lineage-based verification algorithm

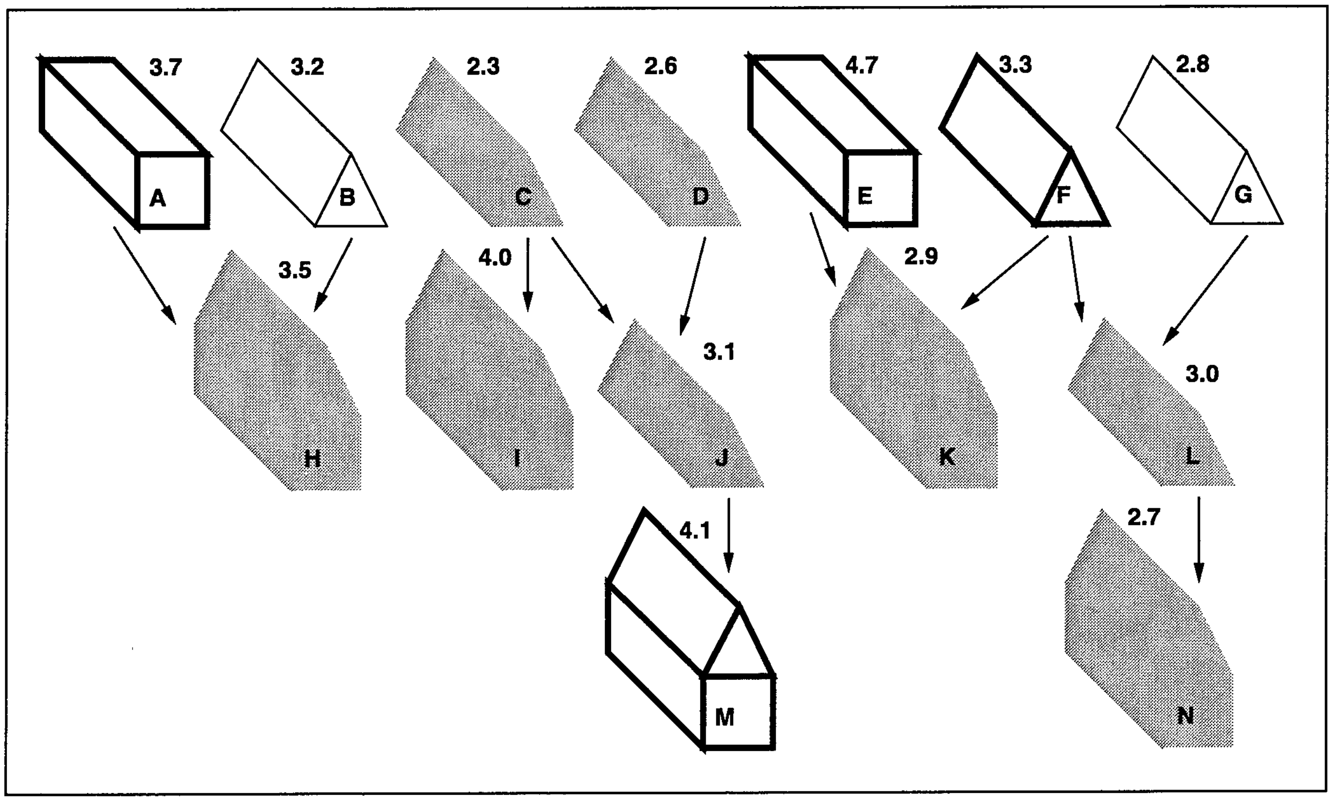


Figure 5-18: Fourth iteration of lineage-based verification algorithm

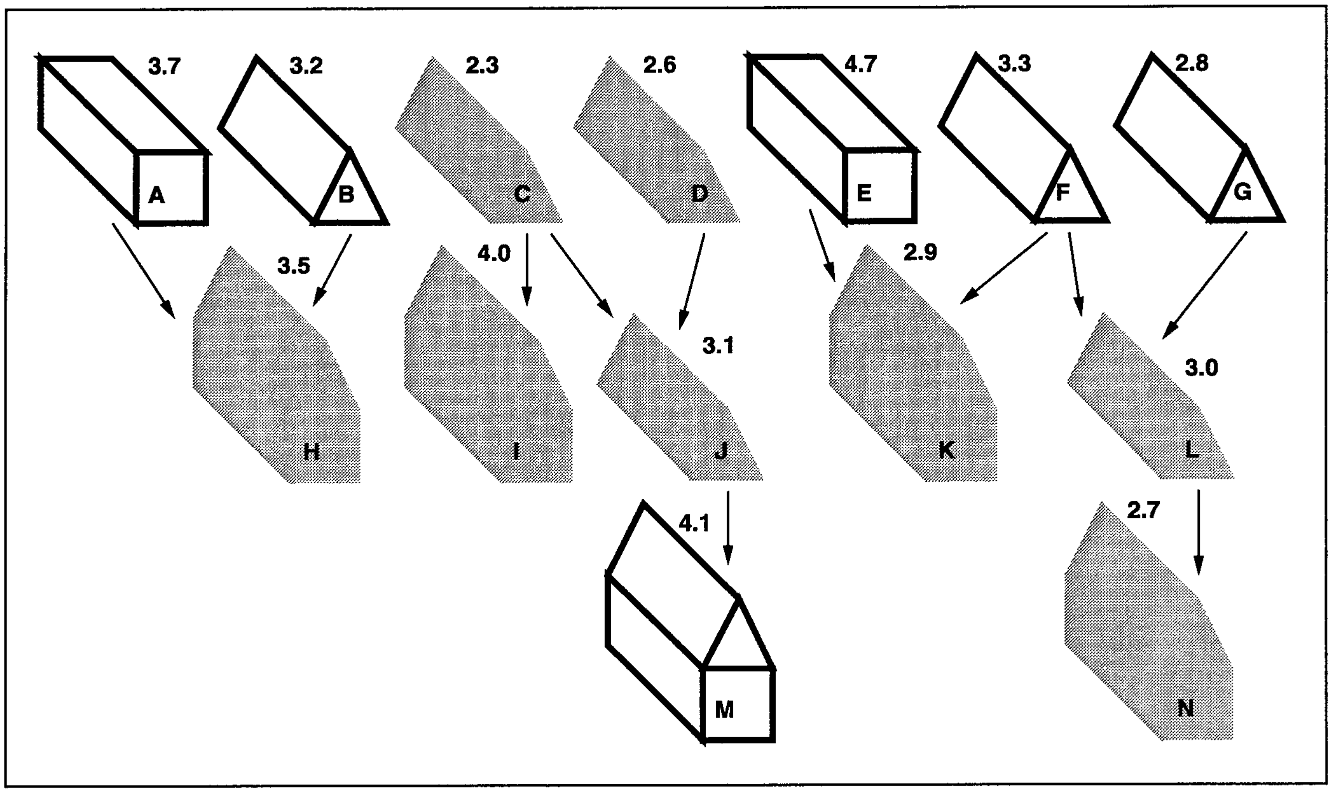


Figure 5-19: Final result of lineage-based verification algorithm

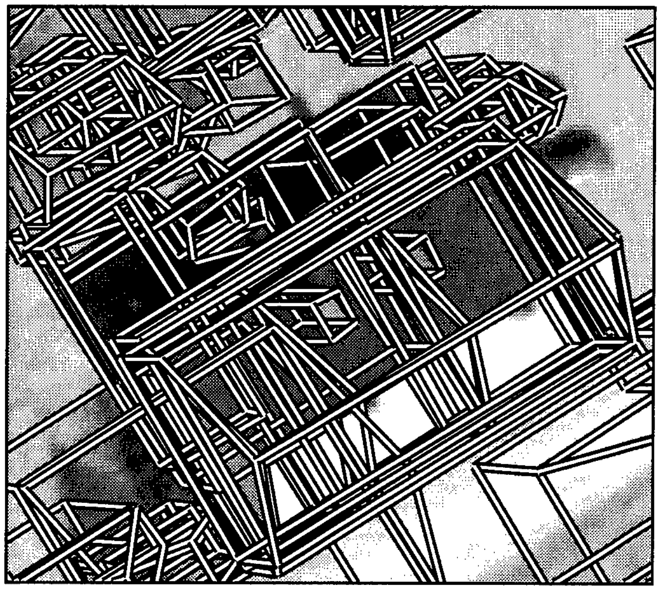


Figure 5-20: Hypotheses prior to verification

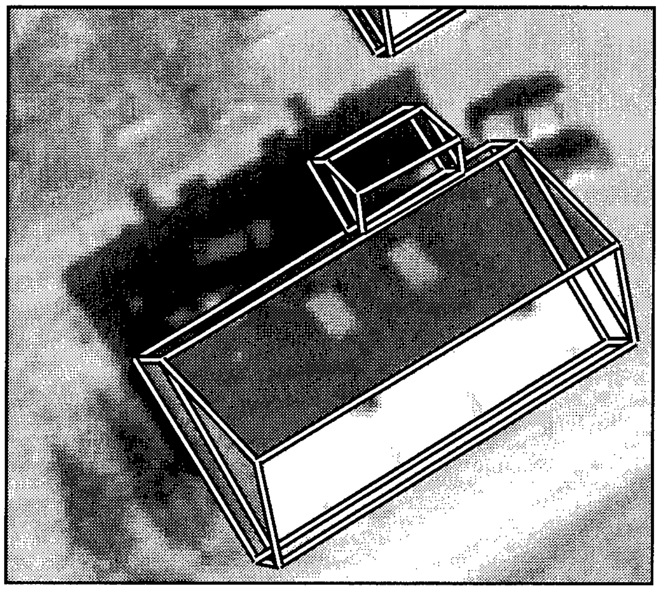


Figure 5-21: Final result

Figure 5-17 shows the result of the third iteration of the verification algorithm. Hypothesis A has the highest score of the remaining active hypotheses, so it is retained. Since H was constructed by attachment of A and B, it is eliminated. However, B is not in direct competition with A, so it remains in the list of active hypotheses.

Figure 5-18 shows the result of the fourth iteration of the verification algorithm. Hypothesis F has the highest score of the remaining active hypotheses, so it is retained. Since L was formed by attachment of F and G, and N was formed by extrusion of L, both L and N are discarded since F had a higher score by itself than as a component of L or N.

Figure 5-19 shows the final result of the verification algorithm. B and G were selected in the last two iterations of the algorithm, since no other competing hypotheses remained in the active list. This example illustrates how a set of object hypotheses can rapidly be pruned by careful use of lineage information, given a reasonable metric for evaluating the quality of each hypothesis. In PIVOT, C_{HYP} serves as such a metric. Again, it is important to stress that this metric is not intended to give any kind of absolute image-independent measure of hypothesis quality; it is instead meant to give a relative measure of quality for hypotheses which share underlying image features.

Next, PIVOT subjects the surviving hypotheses to *overlap analysis*. In this analysis, the remaining hypotheses are again sorted by decreasing confidence score, and in each iteration, PIVOT selects the highest-scoring hypothesis in the sorted list, discarding any other hypotheses whose floor polygons, or *footprints*, overlap with the selected object's footprint. This overlap test is similar in spirit to the *containment analysis* used by Lin and Nevatia [55]; it differs in that it handles partial and complete overlaps, while their analysis dealt only with complete overlap (total containment).

PIVOT's last processing step is to discard any remaining building hypotheses which have very poor image support, or are too small to be considered reliable. If a hypothesis has $E_{HYP}<0.5$, it is discarded, as values below this correspond to very weak or non-existent image gradient at hypothesized object boundaries. If the height of a building hypothesis is less than 1 meter, or has a length or width less than 2 meters, it is discarded. Finally, for peaked roof buildings, two more object-space size constraints are used. If the height of the peak is greater than five times the height of the supporting `rect`, the building is discarded; or, if the supporting `rect` is taller than 10 meters, and the peak height is greater than half of the `rect` height, the building is discarded. These last two constraints have the effect of removing "church steeple" buildings, where spurious slant vanishing points with steep angles were generated in the vanishing point detection process.

To complete the running example introduced at the beginning of this chapter in Figure 3-5, Figure 5-20 shows the entire set of hypotheses, including those produced by combination and extrusion. Figure 5-21 shows the hypotheses remaining after the completion of the verification process. In addition to the correct building model, one spurious hypothesis passes through the verification phase, formed by the alignment of a shadow edge and the edge of a vehicle parked in the shade.

This running example serves as an illustration of the power of primitive combination, extension and verification techniques. Despite edge fragmentation (Figure 5-2) leading to the generation of fragmented primitives (Figure 5-5), the absence of visible verticals necessitating shadow mensuration (Figures 5-10 and 5-11), and the large numbers of geometrically plausible primitives in close proximity to the building (Figure 5-20), PIVOT is still able to extract the correct model and discard the vast majority of incorrect models. This capability comes from its use of primitives as building blocks for object models (Principle 2), modeling photometric effects in 3D (Principle 5), and the use of a rigorous camera model to relate image and object space features (Principle 1).

5.6. General aspects of primitive manipulation and verification

The methods for primitive combination and extension presented in this chapter, along with the lineage-based verification mechanism and its component photometric evaluation techniques, give PIVOT significant leverage for handling the effects of low-level edge fragmentation on hypothesis generation, the effects of viewing angle on the visibility of features such as vertical lines, and the analysis of shadows and illumination for object height estimation and verification. Figures 5-22 and 5-23 show the final PIVOT results for larger areas of the scenes depicted in the running examples of this chapter and Chapter 4; Figures 5-24 and 5-25 show the 3D object space models for each of these images, respectively. Recall once again that these scene models were generated automatically, with no manual intervention, from a single aerial image.

Although PIVOT performs quite well on these images, it is important to note that this initial implementation leaves open several research questions. First, recall that the attachment algorithm for primitives requires that all corner points for two attachable faces must lie in close proximity. One can imagine a large rectangular building with several smaller rectangular wings protruding from one wall. In this case, potentially none of the corner points are in close proximity, and yet the primitives should be attached, as they are all part of the same structure.

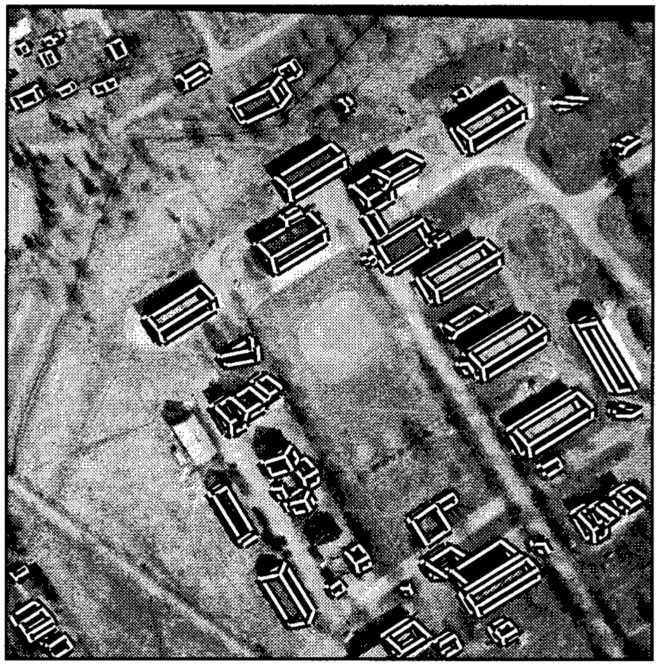


Figure 5-22: Final results for FLAT_L image

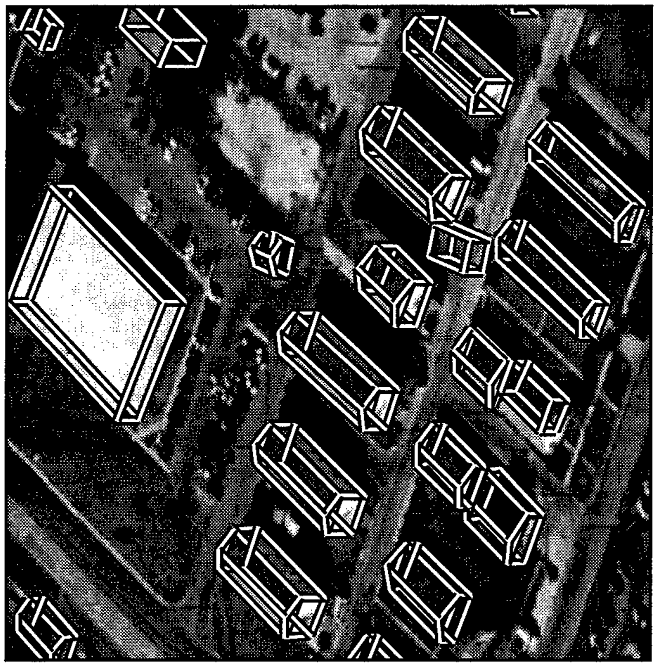


Figure 5-23: Final results for RADT5WOB image

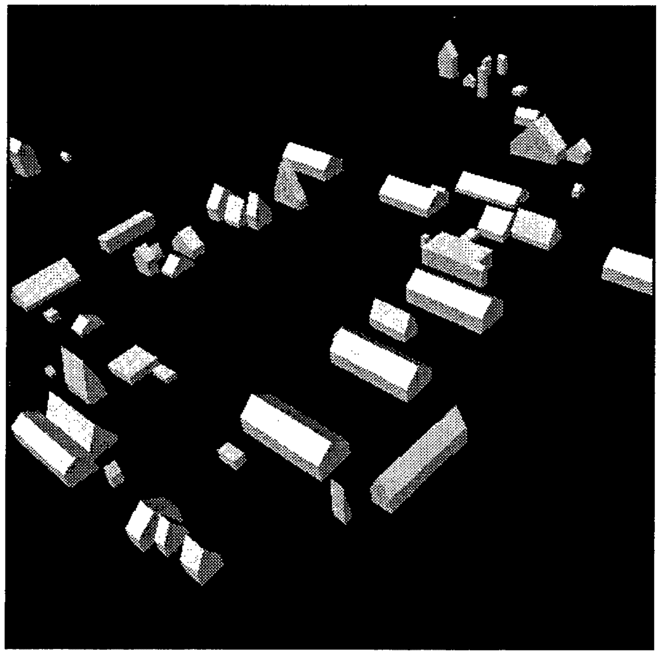


Figure 5-24: 3D object space models, FLAT_L

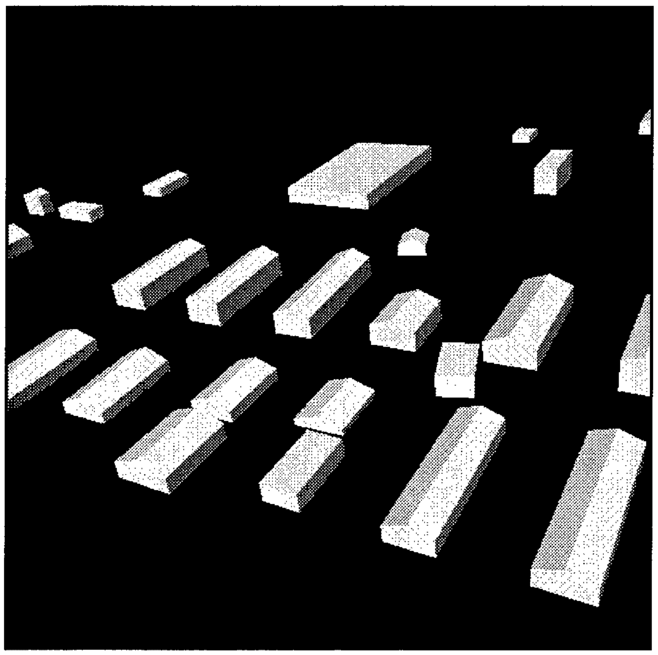


Figure 5-25: 3D object space models, RADT5WOB

Generalizing the attachment algorithm to handle these kinds of situations, as well as arbitrary numbers of attachments instead of the pairwise attachments of PIVOT, remains an open question.

Another question revolves around the interaction between the attachment algorithms and overlap analysis. Consider a building composed of a stack of successively smaller boxes. Since the attachment algorithm is unable to stack these boxes, PIVOT will treat each box as though it were resting on the ground plane. Since the footprints of each box will then overlap, at most one of the boxes will survive the verification process. It should be clear that in the general case, overlap analysis should only be performed after all plausible attachments and extrusions have been formed.

The extrusion algorithms used by PIVOT operate only in the vertical direction. While this is a reasonable design choice, considering the prevalence of structures which can be modeled by vertical extrusions, it is natural to consider extrusions in other directions, including horizontal and slanted directions. One possibility here is to extend the vertical line-based extrusion algorithm described in Section 5.4.1 to extrude along every direction for which a vanishing point has been found. This would give the extrusion algorithm a great deal of power in handling even severe low-level edge fragmentation, but at the expense of generating potentially immense numbers of new hypotheses.

Another issue concerns the implicit and explicit assumptions about intensity coherence on object surfaces. The use of a surface homogeneity term in the C_{HYP} confidence function explicitly rewards objects with smooth intensity surfaces, and the illumination consistency algorithm implicitly makes the same reward by using the median intensity as a representative for surface intensity. Since roofs can be composed of multiple patches of different materials, each with different reflectance properties, and since walls often contain windows and/or entrances, these functions can penalize essentially perfect object models. Texture analysis might be used to identify regular surface markings, such as windows, which could then be ignored during surface evaluation. Feature-based grouping mechanisms could be employed to handle changes in roof markings; Irvin and McKeown [46] used a shadow-based grouping mechanism to handle hypothesis fragmentation caused by abrupt changes in roof material.

Nevertheless, even with the simple combination, extension, and photometric evaluation algorithms described in this chapter, PIVOT can achieve good performance on difficult aerial images, as Figures 5-22 and 5-23 show. However, to fully understand the strengths and limitations of the object detection and delineation algorithms employed by PIVOT, a thorough performance analysis is necessary, covering a wide variety of imagery and comparing PIVOT's performance with existing systems. Chapter 6 presents such an analysis.

Chapter 6
Performance Evaluation and Analysis

In Chapter 2, the computer vision problem was defined in terms of three entities; the scene, image, and scene model. The image acquisition process produced an image of the scene, and the vision system's task was to produce a scene model from the image which matched a manually constructed scene model for the same scene as closely as possible. The issues involved in performing a meaningful comparison of these two kinds of scene models and analyzing the results of such a comparison form the central topics of this chapter, along with a thorough comparative performance evaluation of PIVOT.

Principle 6 argued for fair and comprehensive performance evaluation of object detection and delineation systems. In particular, it was argued that system evaluation should be performed using metrics which present an unbiased picture of system performance, on a large and varied set of imagery. The use of this approach ensures that the performance findings are both statistically significant, in terms of the number of test images used, and representative of system behavior over the entire problem domain, in terms of variations in scene content and image acquisition parameters.

The motivation for this principle, as given in Chapter 2, was the relative lack of useful performance evaluation in previous research in object detection and delineation for aerial imagery. Frequently, the output of these systems is only presented graphically without recourse to quantitative evaluation metrics, and in cases where the scene models are compared with ground truth models, the metrics used give a biased or unclear measure of system performance. Although some work in the literature has been accompanied by thorough and rigorous performance evaluation, this has been the exception rather than the rule.

This chapter begins with a detailed discussion of the issues involved in selecting unbiased and meaningful performance metrics, and the process by which these metrics are used to evaluate PIVOT's scene models. A description of the issues and processes in compiling ground truth scene models is followed by a comparative performance evaluation of PIVOT with two existing monocular building extraction systems on 83 test images covering a wide variety of viewing angles and scene content. While 83 test images cover only a small fraction of the space of possible aerial scenes, it is still possible to evaluate the effects of certain well-represented image and scene properties on the performance of object detection and delineation algorithms.

An analysis of these results is then used to highlight strengths and weaknesses of various object detection and delineation approaches, in the context of image- and scene-related factors which can impact system performance. In particular, experiments are performed to assess the effects of image obliquity, scene complexity, and object density, as well as case studies which illustrate the effects of edge fragmentation, edge grouping, and object orientation on system performance. The analyses presented here represent the most extensive evaluation of automated building extraction systems done to date.

6.1. Selecting evaluation metrics

The performance of an object detection and delineation system can be characterized along several dimensions; many of these dimensions reflect external factors imposed on the system by particular applications. For example, if an object delineation system is intended to be run in a batch mode, followed by manual editing to correct mistakes, then one evaluation measure might be "ease of correction." If some building extraction system A correctly delineates 90% of a building boundary, and requires 10 minutes of manual editing to achieve a perfect delineation, while another system B correctly delineates only 50% of a building boundary, yet requires only 30 seconds of manual editing to achieve a perfect delineation, then the "ease of correction" measure might well favor B, even though its raw building delineations are worse than those of A.

Of course, the "ease of correction" measure is based on an implicit assumption that object delineation must be perfect, and performed as quickly as possible subject to that constraint. In other scenarios, detection might take precedence over delineation. For example, if an object detection system is intended to coarsely locate building structures for rapid population of a cartographic database, then a relevant evaluation measure might be "area mapped per unit time at detection rate x." Such a measure obviously places its emphasis on different aspects of an object detection and delineation algorithm than the "ease of correction" measure.

For the purposes of this work, the performance evaluation focus will be on detection and delineation performance. Stated another way, we wish to determine how closely the scene models generated by PIVOT match "perfect" scene models of the same scenes, by measuring how well they agree on their respective classifications of image space into "object" and "non-object" categories. While it could be argued that this choice of focus is also influenced by an external factor (cartographic accuracy), it can also be argued that measures of detection rate and delineation quality are implicit in the previous examples, and hence represent key aspects of performance that must ultimately be evaluated, independent of external factors or intended applications.

In the evaluation of detection and delineation performance, we seek metrics which are unbiased; in particular, the metrics should not be based in part or in whole on subjective measures of performance, nor should they give results which are subject to interpretation. For example, one might count, manually or automatically, the fraction of buildings produced by the automatic system which overlap buildings in the ground truth scene model, as a measure of detection rate. This is a subjective measure, however, as the degree of overlap necessary for an automatically generated model to overlap a ground truth model is dependent on the human evaluator's judgment or on an arbitrary overlap threshold or matching criterion used by an automatic evaluator. This particular evaluation measure is also subject to varying interpretations; for example, assume that the overlap criterion is 60% area coverage in image space. A system could achieve a 100% detection rate under this measure, yet the actual delineations of objects could cover as little as 60% of the ground truth scene model.

The evaluation metrics used in this work have been described and employed elsewhere [55, 67, 92, 93], although the specific definitions used in previous work have varied; the definitions used uniformly throughout this work are consistent with those given by Shufelt and McKeown [93] and are described in detail here to avoid confusion. The metrics are based on classifications of pixels in image space and voxels in object space by the object detection and delineation system's scene model and by the ground truth scene model. A scene model classifies each pixel in an image, and each voxel in object space, into one of two categories: object or non-object (background). Since there are two scene models in question, there are four possible categories for each pixel (or voxel):

- **true positive (TP)**: both the vision system's scene model and the ground truth scene model classify the pixel (voxel) as belonging to an object.
- **true negative (TN)**: both the vision system's scene model and the ground truth scene model classify the pixel (voxel) as belonging to the background.
- **false positive (FP)**: the vision system's scene model classifies the pixel (voxel) as belonging to an object, but the ground truth scene model classifies the pixel (voxel) as belonging to the background.
- **false negative (FN)**: the vision system's scene model classifies the pixel (voxel) as belonging to the background, but the ground truth scene model classifies the pixel (voxel) as belonging to an object.

To evaluate performance, the number of TP, TN, FP, and FN pixels and voxels are counted, and then the following metrics are computed, once for image space and once for object space:

- **building detection percentage**: $\frac{100\,TP}{TP+FN}$
- **branching factor**: $\frac{FP}{TP}$
- **quality percentage**: $\frac{100\,TP}{TP+FP+FN}$

The *building detection percentage* is the simplest metric, measuring the fraction of ground truth pixels which were correctly denoted as object pixels by the vision system. The *branching factor* is a measure of the degree to which a system overclassifies background pixels as object pixels. If a system never "overdelineates" the extent of any object, its branching factor would be zero. A system with a branching factor of one would incorrectly label a background pixel as an object pixel for every object pixel it correctly detected; the best possible branching factor would be zero. Finally, the *quality percentage* measures the absolute quality of the vision system's scene model. To obtain 100% quality, the vision system must correctly label every object pixel, without missing any (FN), and without mislabeling any background pixels (FP). In other words, the vision system must produce a perfect classification of the image with respect to the ground truth to obtain a quality percentage of 100%.

The building detection percentage can be treated as a measure of object detection performance; the branching factor can be treated as a measure of delineation performance. The quality percentage combines aspects of both measures to summarize system performance. All three measures taken together give a clear picture of system performance with simple and unambiguous interpretations and without any subjective elements.

To be sure that the reader understands these metrics, a brief example follows. Consider an image with 1000 pixels, in which a single building covering 300 of the pixels is present, leaving 700 background pixels. Further assume that a vision system X correctly labels 240 of the 300 object pixels, and 560 of the 700 background pixels. So, there are 240 TP pixels, 560 TN pixels, 140 FP pixels, and 60 FN pixels. It follows that the building detection percentage in this example is 80%, the branching factor is .583, and the quality percentage is 54.5%.

The image space versions of these metrics are implemented by polygon scan conversion into a collector image with the same pixel dimensions as the vision system's input image, from which the counts of TP, TN, FP, and FN pixels can be easily obtained. The object space versions can be implemented by testing voxels in object space for overlap with polyhedra in the scene models, and counting TP, TN, FP, and FN voxels. The resolution of the voxel grid should be chosen so that it is comparable to the effective image resolution and significantly smaller than the size of detectable objects; for the object space evaluations on the aerial imagery used in this work, cubic voxels $0.5m$ on a side were used.

6.2. Ground truth scene model compilation

Our goal in performance evaluation is to compare the scene models produced by PIVOT for a given set of aerial imagery with the most accurate and precise scene models which can be generated from the same set of imagery. In practice, scene models which meet these requirements are compiled by manual measurements of image points, combined with photogrammetric triangulation of these points to produce 3D wireframes in object space.

For the evaluations presented in this work, ground truth scene models were compiled with the use of SITECITY, a semi-automated multi-image site modeling system developed at the Digital Mapping Laboratory at Carnegie Mellon University. SITECITY combines interactive measurement tools, image understanding techniques, and a rigorous constrained least-squares photogrammetric bundle adjustment package [66] for site model generation. Thorough descriptions of this modeling system and detailed usability evaluations appear elsewhere [40, 41]. To give a rough idea of SITECITY's capabilities, Figure 6-1 shows the image measurements for the NEST_FHN711 scene, covering a portion of Fort Hood, Texas. Figure 6-2 shows the object space model triangulated from six overlapping images, taken from different viewpoints, of the same area. This example shows SITECITY's ability to model a wide variety of building shapes; the peaked roof buildings have overhangs, the main building has several layers tied together by coplanarity constraints, and the curved strip on the front of the main building is itself an overhang, with a rectangular hole.

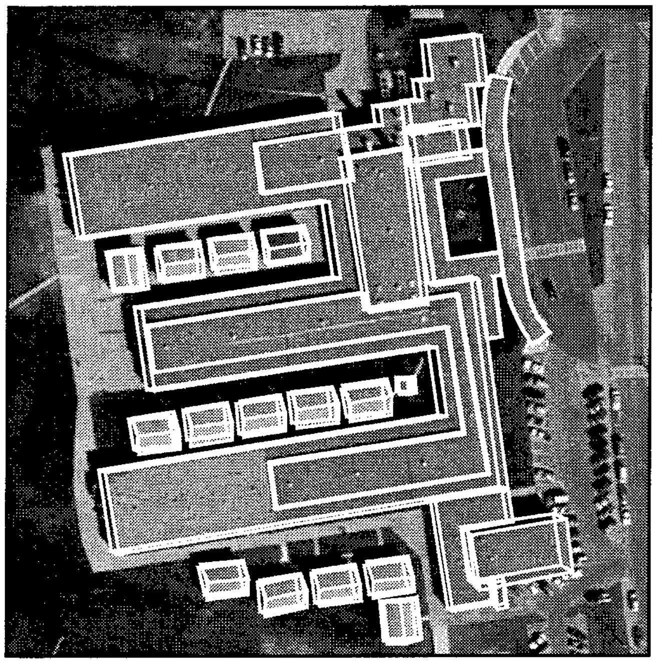


Figure 6-1: SITECITY measurements on NEST_FHN711

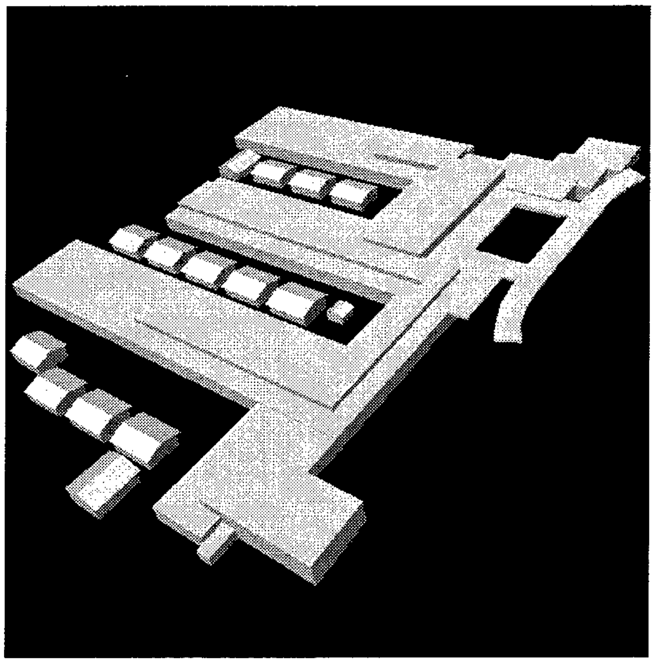


Figure 6-2: Object space model for NEST images

Scene models for the 83 test images used in this work were measured by four people, to guard against measurement bias which might result from the use of one individual's measurements exclusively. SITECITY's rectilinear building models use internal constraints to maintain right angles at corners vertical walls, and horizontal floors and roofs; for complex multi-level structures, aligned building faces, and atypical building shapes, constraints (collinearity, coplanarity, and right-angle constraints) were added as necessary to the building models to obtain geometrically consistent solutions.

In addition to the topology of the building wireframes, SITECITY maintains the measured image coordinates of each object space point. This raises an issue for 2D image space evaluation: which set of 2D ground truth building delineations is most appropriate for comparison with PIVOT's building delineations? To understand the various possibilities, consider a scene A, of which n images A_1–A_n, taken from different viewpoints, are available. For image A_i, there are five distinct methods for generating 2D delineations from SITECITY-compiled ground truth:

1. Use the manually generated building measurements for image A_i.

2. Project the manually generated building measurements for image A_i through the camera model to object space, using a digital elevation model (DEM) to obtain floor point elevations and measuring object space dimensions of model edges in the image given their expected orientations in object space. Project the resulting object space model back to image space.

3. Project the manually generated building measurements for image A_i through the camera model to object space, using a DEM to obtain floor point elevations and measuring object space dimensions of model edges in the image. Fit the resulting object space model to an idealized object model, and project the idealized model back to image space.

4. Project the manually generated building measurements for all n images through the camera model to object space, and project the resulting object space model back to image space.

5. Project the manually generated building measurements for all n images through the camera model to object space, fit the resulting object space model to an idealized object model, and project the idealized model back to image space.

Methods 2 and 3 can be distinguished from 4 and 5 by the way in which 2D image points are converted to 3D object space points. In methods 2 and 3, since only one image is being used, the elevations of floor points must be obtained by intersecting a ray from the perspective center through the image point with a DEM. The distances along horizontal edges connecting floor points and the heights of vertical edges connecting floor points to roof points can be computed with photogrammetric methods using the camera model, much in the same manner that PIVOT generates 3D object space models from its monocular 2D primitive wireframes. In contrast, methods 4 and 5 use multi-image triangulation of image measurements to obtain 3D points. Since DEMs typically have coarse resolution compared to frame mapping photography-based measurements, there will be a discrepancy in the position of building models computed by methods 2 and 3 with those computed by methods 4 and 5.

Methods 2 and 4 can be distinguished from 3 and 5 by the latter pair's use of idealized building models. Due to cumulative errors in manual image measurements and the camera model resection, it is rare that the initially computed 3D point positions give geometrically clean models; expected right angles are often acute or obtuse, building faces which are expected to be parallel are often skewed, and expected verticals exhibit tilt. Methods 3 and 5 use a least-squares solution to fit the initial 3D point positions to a geometrically consistent building model. In method 5, the fit is included in the bundle adjustment for point position triangulation; since method 3 only uses one image, the fit is done after the computation of 3D point positions.

Since the goal is to compare PIVOT's scene models with the most accurate scene models possible, method 1 is used for 2D evaluation. The manual image measurements represent the closest possible fit to corner points and edges, so any discrepancies between PIVOT's results and the ground truth can be attributed solely to delineation inaccuracies in PIVOT. Using any of the remaining methods introduces other sources for delineation error, and unnecessarily complicates the 2D performance analysis.

To evaluate PIVOT's 3D object space models, a hybrid approach is required. On one hand, the object space models with the best accuracy and precision come from multi-image constrained triangulation, which would suggest the use of method 5. On the other hand, since PIVOT is a monocular system, it must obtain point elevations with a DEM, and hence comparison of PIVOT's 3D wireframes with the ground truth models of method 5 will introduce positional error in the evaluation due to the discrepancy between multi-image triangulation and ray intersection with a DEM. The solution is to first compute 3D wireframe ground truth models using method 5, to obtain the best solutions for building shape and extent. Then, the image measurements for image A_i are used with method 2 to obtain a building position with the DEM. The translation between this single-image building solution and the multi-image solution of method 5 is computed, and this translation is applied to the multi-image models to bring them into alignment with the DEM. This eliminates positional discrepancies, and any remaining error can be attributed solely to PIVOT.

Before continuing with a discussion of the performance evaluation methodology, it should be noted that even the ground truth scene models compiled here for evaluation are still subject to slight errors in manual image measurements, as are all scene models which have been compiled manually with image-based modeling tools. In SITECITY experiments with 12 subjects, the average measurement uncertainty was found to be on the order of one pixel [40]. However, given the size of buildings in the test images for these experiments, and the relative magnitude of delineation errors made by automated systems, ground truth uncertainties of this size can be safely ignored. When automated monocular or multi-image analysis systems achieve single-pixel or sub-pixel precision, it will become necessary to maintain covariance information on manually compiled ground-truth models, and use this information in performance evaluation.

6.3. Comparative performance evaluation methodology

A particularly important aspect of performance evaluation is the ability to compare performance measurements across multiple object detection systems. There exist a multitude of factors which can affect the performance of these systems; understanding why certain systems exhibit robust detection behavior on certain kinds of imagery, while others suffer performance breakdowns, gives insight into the specific factors that affect object detection and delineation algorithms.

In addition to a thorough performance analysis of the PIVOT system, a comparative performance analysis is done with three other building extraction systems from the literature, each with different underlying assumptions about the imaging process and scene model structure. By comparing performance on a varied test set of imagery across these four systems, strengths and weaknesses of the detection and delineation algorithms can be ascertained. To understand the essential differences between these systems and PIVOT, a short description of each of the three earlier systems follows. As with PIVOT, all three systems run without manual intervention on a single image. All three systems also use the Nevatia-Babu edge operator [78] to extract raw edge data.

- BUILD [68] is a line and corner-based analysis system which operates solely in image space. BUILD assumes that all images are acquired with nadir or near-nadir imaging geometry, that perspective effects can be ignored, and that all buildings can be modeled by 2D convex quadrilaterals (also known as *boxes*). BUILD begins by using a sequence finder [110] to break edges at points of high curvature, and then uses a collinear line linking process to handle fragmented edges which appear to share the same underlying structure.

  BUILD then uses a 2D range search to find pairs of edges which meet at approximately right angles. The resulting corners are used to form a line-corner graph, from which chains of corners with the same rotation are extracted to form boxes. If no chain can be found for a corner, BUILD uses symmetry to generate a corner. BUILD assumes the solar azimuth angle is known, and uses that information to calculate the sides of the boxes which are expected to cast shadows. A statistical analysis of the shadow-casting sides of the boxes is then used to determine a global shadow intensity threshold, and boxes which have strong shadow support are selected as the final 2D hypotheses. By itself, BUILD does not generate 3D building hypotheses.

- BUILD+SHAVE is a hybrid system comprised of BUILD and a shadow-based verification system, SHAVE [46]. Like BUILD, SHAVE assumes that all images are acquired with nadir or near-nadir imaging geometry, and that perspective effects can be ignored. SHAVE uses the global shadow threshold computed by BUILD in conjunction with a sequence finder [110] to delineate shadow regions on the shadow-casting sides of boxes, using the known solar azimuth. After delineating a shadow region, SHAVE computes the average length of the shadow region, which could be used to derive an image space estimate of building height in conjunction with the solar elevation angle.

  BUILD+SHAVE simply runs BUILD to produce 2D boxes, and then runs SHAVE on those boxes to obtain shadow lengths for each box. The ground sample distance is computed at the center of the box, and this is multiplied by the length of the shadow in image space to obtain the length of the shadow in object space, which can then be used with the solar elevation angle to derive an object space height estimate for the building. The photogrammetric routines described in Section 5.3 are then used to generate a 3D object space wireframe model from the 2D box and height estimate.

- VHBUILD [67] is the successor to BUILD, relaxing the nadir acquisition geometry assumption, and incorporating a rigorous camera model. As with BUILD, VHBUILD models rooftops with boxes, but VHBUILD also handles peaked roof buildings whose roof facets are symmetric rectangles. VHBUILD allows corners to form any angle, unlike BUILD; this increase in the hypothesis search space is countered by using vanishing point constraints to prune corners and boxes, using essentially the same ideas as those in Section 4.2. It should be noted, however, that VHBUILD does not model slanted vanishing points explicitly. It uses only one label to denote all non-horizontal and non-vertical edges. As a consequence, VHBUILD only histograms the Gaussian sphere for horizontal vanishing points.

  VHBUILD uses the same nadir-based shadow verification mechanism as BUILD; the vanishing point labels on the verified boxes are then analyzed to determine whether the box is the roof of a flat roof building or a peaked roof building. VHBUILD then invokes a vertical edge finder on these verified 2D boxes to obtain height estimates for the boxes. These height estimates are then used to generate 3D object space wireframe models.

It is useful to highlight the differences between the four building extraction systems. These differences are summarized in Table 6-1, which lists various dimensions of building extraction systems and describes where each of BUILD, BUILD+SHAVE, VHBUILD, and PIVOT lie with respect to each of those dimensions.

The test data used in these comparative experiments consists of 83 images, covering a variety of geographic areas, viewing angles, illumination angles, and scene complexities. Each scene occurs at least twice in the test data, covered from at least two different viewpoints. Details of the test imagery and the actual results for all 83 images across all four building extraction systems, as well as the performance metrics, are presented in Appendix B.

The experimental methodology is simple; each system is run without any parameter adjustment or manual intervention on exactly the same imagery, and no parameter adjustments or modifications are made between runs on different images. The performance metrics described in Section 6.1 are then computed for both the 2D image space and 3D object space results from each system (recall that BUILD generates 2D image space results only, so no 3D evaluation is done for this system).

system properties	BUILD	BUILD+SHAVE	VHBUILD	PIVOT
scene model type	2D image space quadrilaterals	3D object space extruded quadrilaterals	3D object space rectangular and peaked roof building models	3D object space building models composed of rectangular and triangular primitive volumes
vanishing point modeling	none	none	verticals and orthogonal horizontals	verticals, orthogonal horizontals, and pairs of symmetric slanted "peaks"
viewpoint assumptions	nadir acquisition geometry	nadir acquisition geometry	rooftops are visible	none
shadow modeling	nadir-based image space approximation	nadir-based image space approximation	nadir-based image space approximation	full 3D object space shadow modeling
intermediate feature requirements	one corner (2 edges) necessary for hypothesis generation	one corner (2 edges) necessary for hypothesis generation	one corner (2 edges) necessary for hypothesis generation	one 2-corner (3 edges) necessary for hypothesis generation
height measurement capabilities	none	shadow-based	vertical-based	shadow- and vertical-based
preliminary edge processing	collinear line grouping	collinear line grouping	collinear line grouping	recursive line splitting

Table 6-1: Summary of system differences

6.4. Baseline performance results and comparative analysis

To begin the comparative analysis, it is useful to examine the performance of related pairs of systems. In particular, BUILD and BUILD+SHAVE share the same underlying 2D nadir-based box generation mechanism, so it is natural to ask how the addition of a shadow-based height approximation component affects BUILD's performance in 2D (recall that BUILD only operates in image space, and does not generate 3D building models). In the other case, PIVOT and VHBUILD both employ a vanishing point-based approach for building hypothesis generation; here, it is natural to ask whether PIVOT's more rigorous vanishing point detection mechanism and multiple height measurement methods lead to improved performance over VHBUILD.

Figures 6-3 through 6-5 compare the performance of BUILD against BUILD+SHAVE, using each of the performance metrics described in Section 6.1. Each dot in the graphs represents one of the 83 test scenes.

First, we observe in Figure 6-3 that BUILD+SHAVE outperforms BUILD in building detection rate, for all of the test scenes. This is to be expected, since BUILD+SHAVE starts with the 2D roof boundaries generated by BUILD, and then adds walls based on SHAVE's shadow-based mensuration of building height. Since adding wall polygons to the 2D delineations will increase the number of true positives and decrease the number of false negatives, or increase the number of false positives, the building detection rate will never decrease.

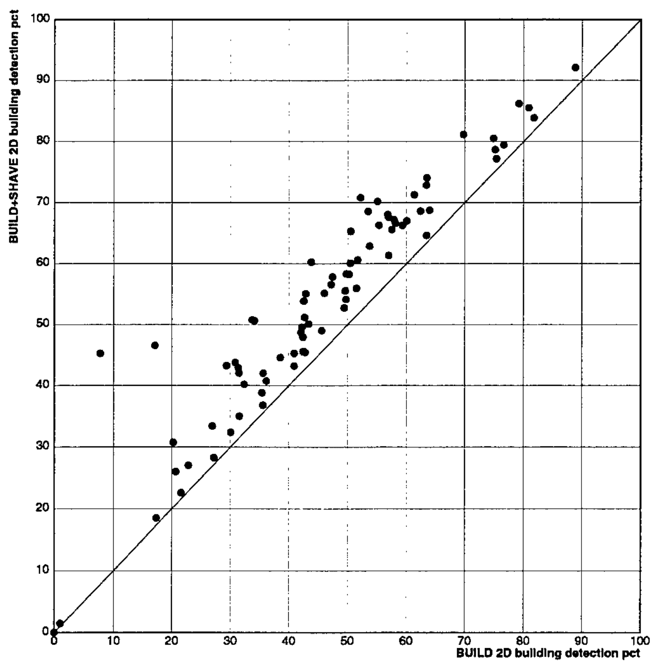


Figure 6-3: 2D building detection, BUILD vs BUILD+SHAVE

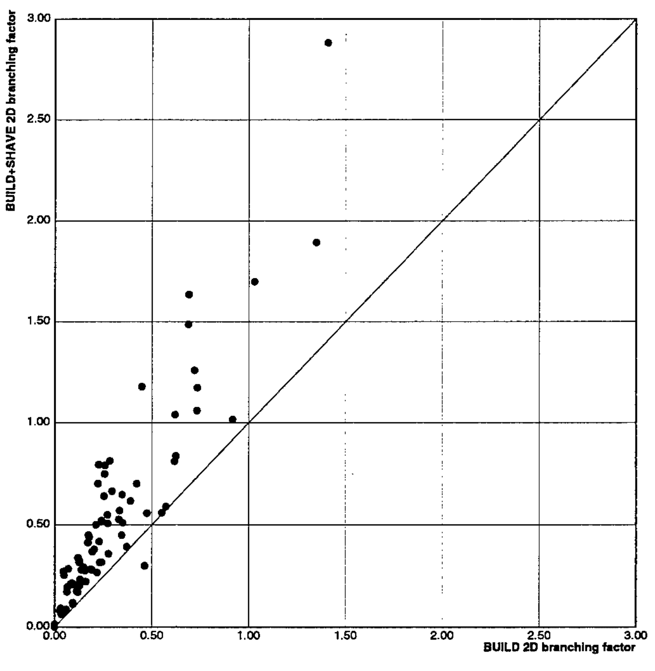


Figure 6-4: 2D branching factor, BUILD vs BUILD+SHAVE

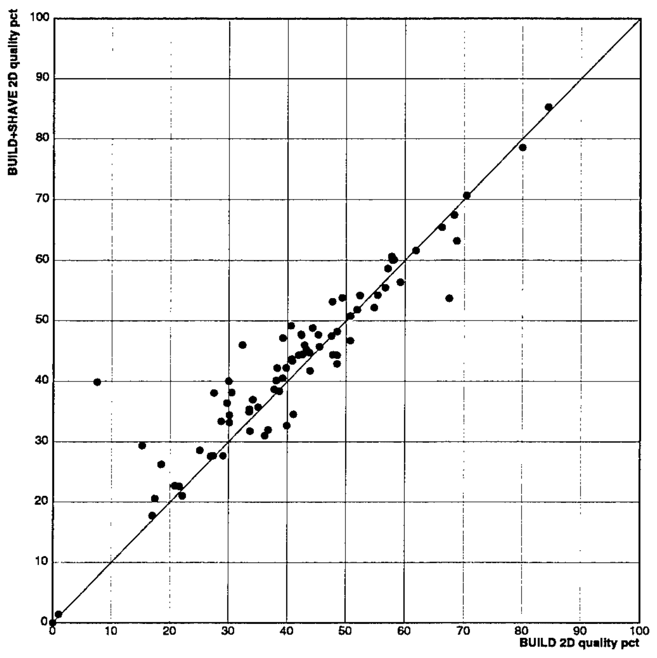


Figure 6-5: 2D quality pct, BUILD vs BUILD+SHAVE

Next, note in Figure 6-4 that the branching factor almost always increases when SHAVE's wall polygons are added to BUILD's roof polygons. This is a consequence of two factors. First, any erroneous BUILD rooftop polygons will now have wall polygons added, increasing the number of false positives without increasing the number of true positives. Second, for good BUILD rooftop polygons, SHAVE's shadow delineation procedure can still overestimate the extent of the shadow boundary, leading to an overestimate of building height, and overextended building walls, which, while contributing to the number of true positives, also contribute to the number of false positives.

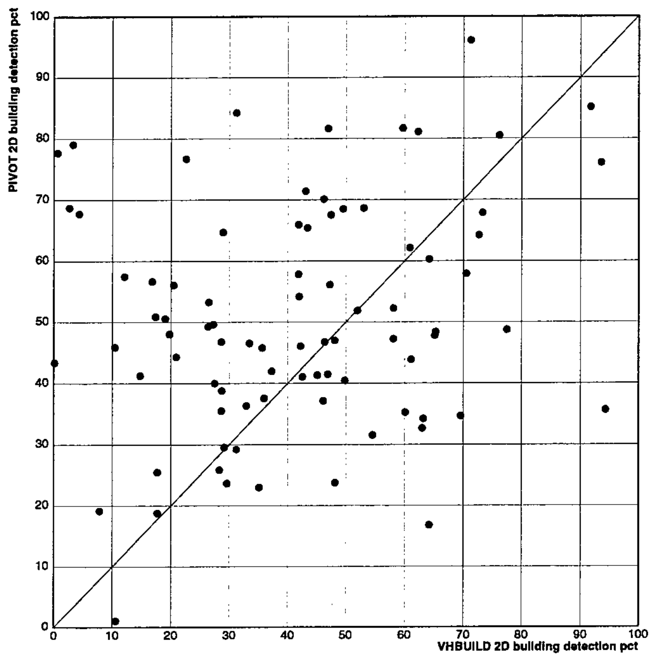


Figure 6-6: 2D building detection pct, VHBUILD vs PIVOT

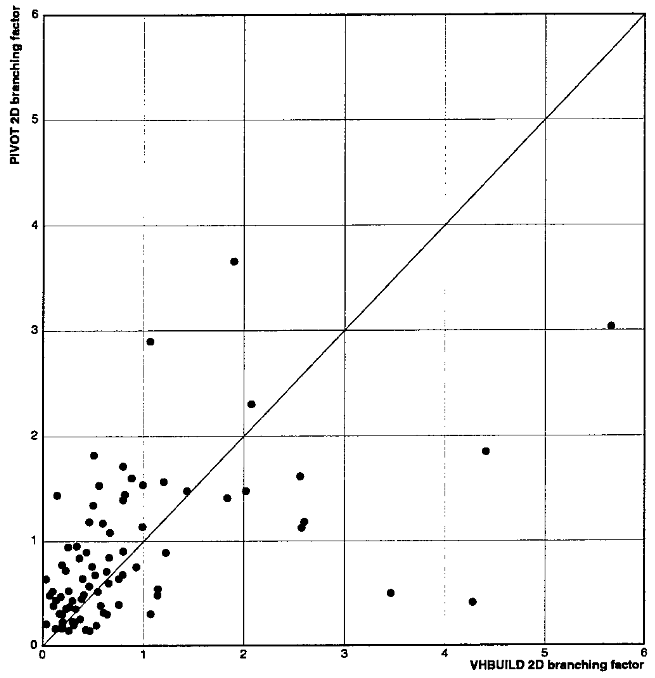


Figure 6-7: 2D branching factor, VHBUILD vs PIVOT

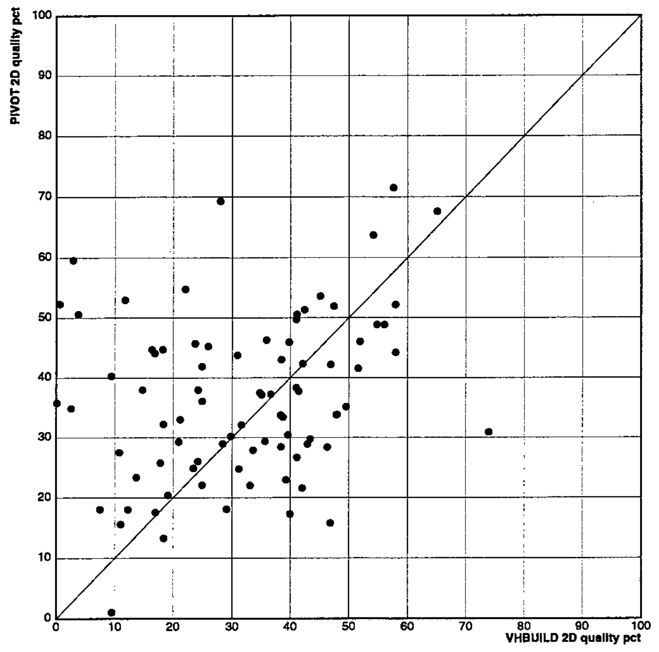


Figure 6-8: 2D quality pct, VHBUILD vs PIVOT

Only one scene (MMBLD-71OV) gains sufficiently many true positives to offset the number of false positives, and as Figure B-142 in Appendix B shows, these true positives are generated by essentially accidental building hypotheses.

Finally, the quality percentage for both systems is plotted in Figure 6-5. Since the increase in detection rate is often offset by an increase in the branching factor, the change in the quality percentage metric is often slight, reflecting the changes in both the number of true positives and in the number of "false" pixels. The results summarized in Figures 6-3 through 6-5 show no significant overall performance difference in terms of 2D building detection and delineation. To simplify the exposition, only BUILD+SHAVE will be considered in successive comparative analyses, as it also produces 3D building models, allowing object space comparisons of performance with the other systems.

Next, we consider the 2D detection and delineation performance of VHBUILD and PIVOT. Figures 6-6 through 6-7 compare the two systems, using the same plotting scheme as in the previous figures.

In contrast with BUILD and BUILD+SHAVE, Figure 6-6 shows little correlation between the performance of VHBUILD and PIVOT. While VHBUILD exhibits significantly better detection performance than PIVOT on several scenes, the converse is also true. Overall, PIVOT has a higher 2D building detection rate in 51 out of the 83 scenes. Similar performance behavior can be seen in Figure 6-8, which illustrates the quality percentage. Here, PIVOT exhibits a higher quality percentage on 50 scenes.

PIVOT exhibits this overall performance increase in part because of its ability to generate more building hypotheses than VHBUILD. The box generation and verification algorithm shared by BUILD, BUILD+SHAVE, and VHBUILD is conservative, in that the boxes it generates must have strong edge and shadow support to be accepted; boxes which have either weak shadow support or weak edge gradient are rejected. PIVOT's hypothesis generation algorithm, on the other hand, only rejects models with weak edge gradient support, while models with weak shadow support can be accepted if they do not overlap or share feature lineage with any model possessing higher verification scores. PIVOT's ability to generate more hypotheses is reflected in the branching factor plots, in Figure 6-7.

The branching factor statistics generally favor VHBUILD, which produces a lower branching factor on 49 of the 83 scenes (two outliers are not shown on this plot; both favor PIVOT). Since PIVOT generally creates more building hypotheses than VHBUILD, it can be expected that the number of "positives" will increase as well, and since some of these will have weak shadow support, the number of false positives may increase at a faster rate than true positives. On the other hand, more hypotheses generally lead to fewer missed buildings, as the building detection statistics show.

While these statistics moderately favor PIVOT, the major distinction between PIVOT and VHBUILD is to be found in the 3D object space performance evaluation statistics. Figures 6-9 through 6-11 present these statistics, again in the same format as previous plots.

First, we examine 3D building detection percentage, in Figure 6-9. In comparison with Figure 6-6, note a general shift of the points to the lower left of the graph. This shift reflects the addition of another source of error, the height measurements for each 2D boundary. Given a 2D building detection rate x, the only way to maintain that rate of building detection performance in 3D is to perfectly measure, or over-measure, the heights of each structure. Any under-measurements of height result in missed building voxels, which translates to reduced building detection performance in 3D.

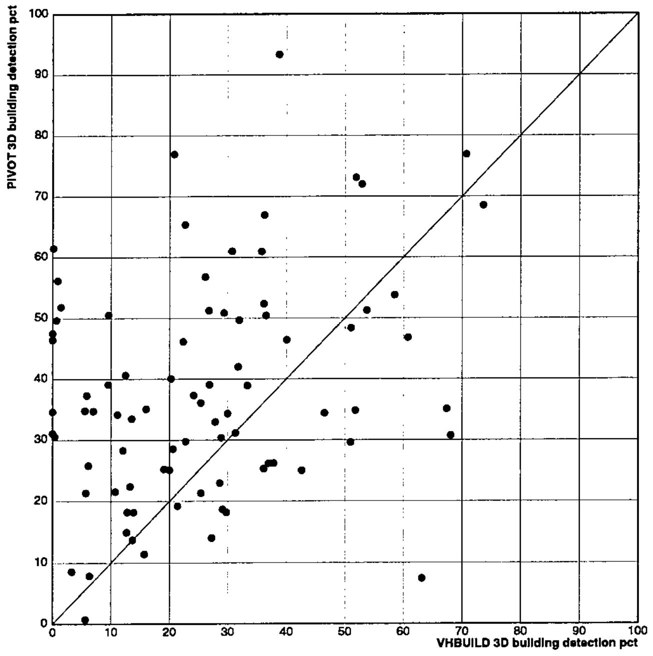


Figure 6-9: 3D building detection pct, VHBUILD vs PIVOT

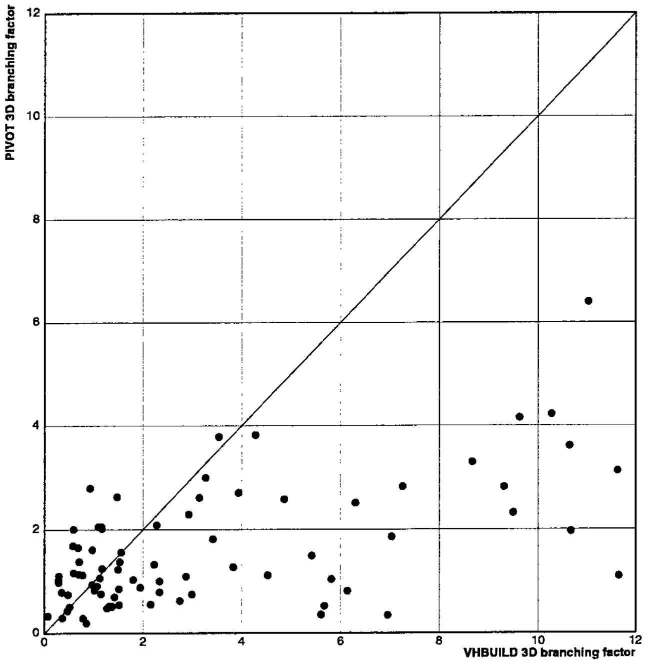


Figure 6-10: 3D branching factor, VHBUILD vs PIVOT

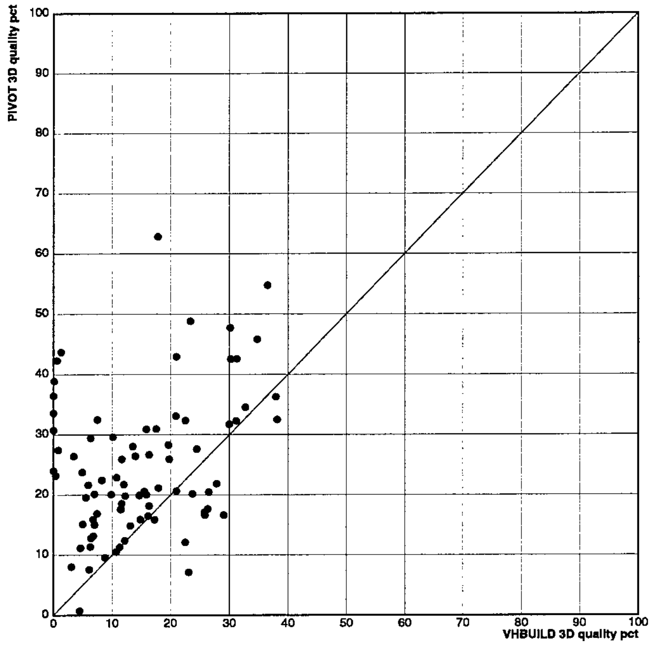


Figure 6-11: 3D quality pct, VHBUILD vs PIVOT

In relative terms, PIVOT's building detection performance over VHBUILD improves in 3D; PIVOT outperforms VHBUILD on 59 of the 83 scenes. This can be attributed to PIVOT's enhanced height estimation capabilities; rather than just using vertical lines to measure height, PIVOT also employs shadow analysis. However, this plot only shows that PIVOT has superior detection performance; it does not indicate the degree to which either system might be over-hypothesizing building structure.

Figure 6-10 shows the 3D branching factor statistics. Here, PIVOT exhibits a superior branching factor on 61 of the 83 scenes (5 outliers are not shown; all of these favor PIVOT). Many of the VHBUILD results exhibit very high branching factors; in 10 cases, VHBUILD is generating over 8 false positive voxels for every true positive voxel it generates. The bulk of PIVOT's results show a branching factor of 2 or less, indicating that while it also over-hypothesizes scene structure, its performance in this category is substantially better than VHBUILD.

Figure 6-11 shows the 3D quality percentages for both systems. Again, comparing this figure with the 2D quality percentages, a shift to the lower left can be observed, for the same reasons as outlined for the building detection percentage. PIVOT exhibits superior 3D quality performance in 67 of the 83 scenes, which is a reflection of the improvements in branching factors (indicating less over-hypothesizing of building structure) and in building detection percentage (indicating less under-hypothesizing of building structure). Since PIVOT showed only moderate average-case improvement over VHBUILD in terms of 2D detection and delineation, the significant improvements in 3D performance can be attributed to the differences in the height measurement algorithms. As noted earlier, PIVOT uses both shadows and verticals to determine height, as opposed to VHBUILD's exclusive use of verticals.

Figures 6-6 through 6-11 show no clear performance *correlations* between VHBUILD and PIVOT. In fact, PIVOT shows no clear performance correlation with BUILD or BUILD+SHAVE, either; the figures for these pairings have been omitted for brevity. Since there are no correlations, it is natural to present the performance data in a different format, which highlights the relative change in performance between VHBUILD and PIVOT.

Figures 6-12 and 6-13 present the data shown earlier in Figures 6-6 and 6-9, respectively, but in a different format. Here, VHBUILD's building detection percentage is subtracted from PIVOT's building detection percentage for each of the 83 test scenes, and the resulting values are sorted in increasing order. Vertical bars connecting each point with the x-axis serve as visual aids. Comparing Figure 6-12 with Figure 6-13, the improvement in PIVOT's performance from 2D to 3D can now be seen as a "shift" of the plot to the left, or alternatively, by the increased area covered by the vertical lines above the x-axis.

Similarly, Figures 6-14 and 6-15 show the relative change in quality percentage from VHBUILD to PIVOT, recapitulating the statistics presented in Figures 6-8 and 6-11, respectively. Once more, the improvement in PIVOT's quality percentage statistics from 2D to 3D can be seen in the increased area covered by the vertical lines above the x-axis.

In the comparisons presented so far, BUILD and BUILD+SHAVE were shown to have comparable 2D performance; BUILD+SHAVE's ability to generate 3D structures led to a higher detection rate, but this was also accompanied by the generation of higher numbers of false positives, leading to roughly the same quality percentage in both cases. For VHBUILD and PIVOT, the 2D detection and delineation algorithms both showed weaknesses and strengths in different parts of the performance space; PIVOT, however, exhibited superior 3D object detection and delineation performance.

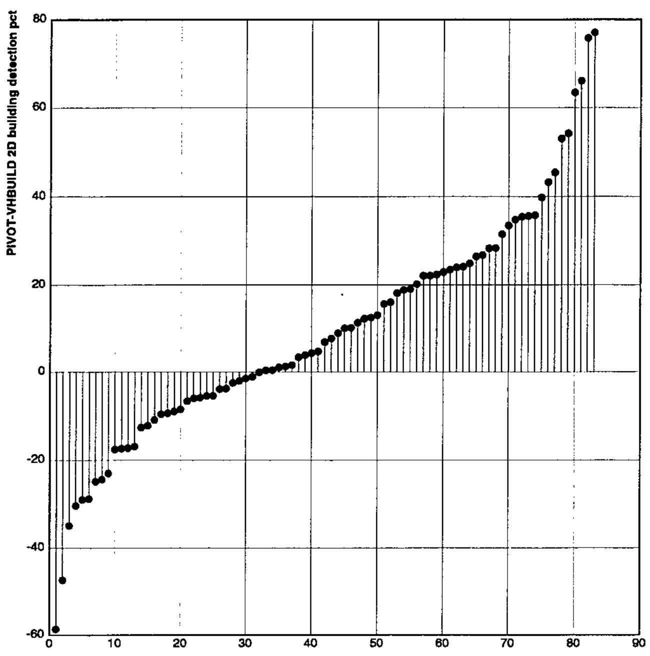


Figure 6-12: Relative 2D bld detection, VHBUILD vs PIVOT

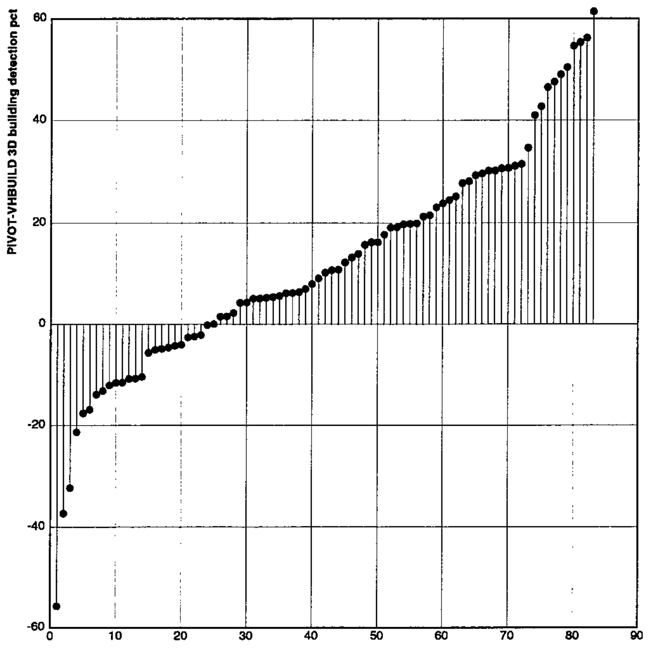


Figure 6-13: Relative 3D bld detection, VHBUILD vs PIVOT

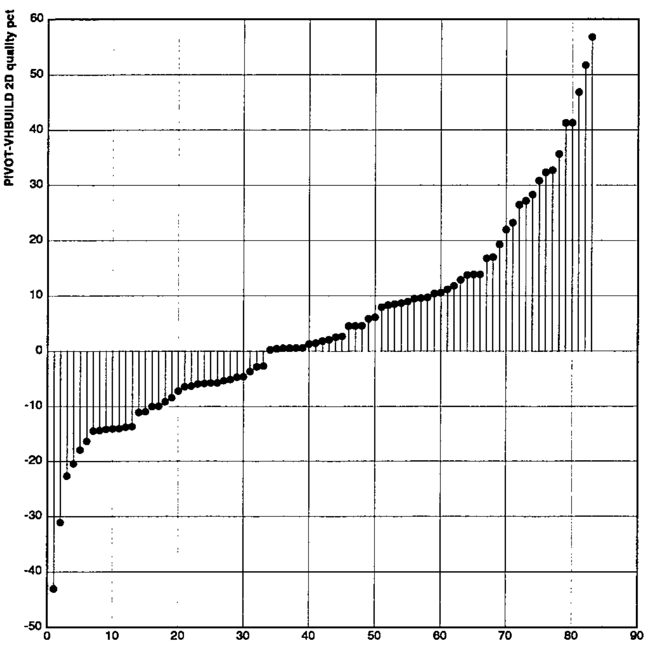


Figure 6-14: Relative 2D quality, VHBUILD vs PIVOT

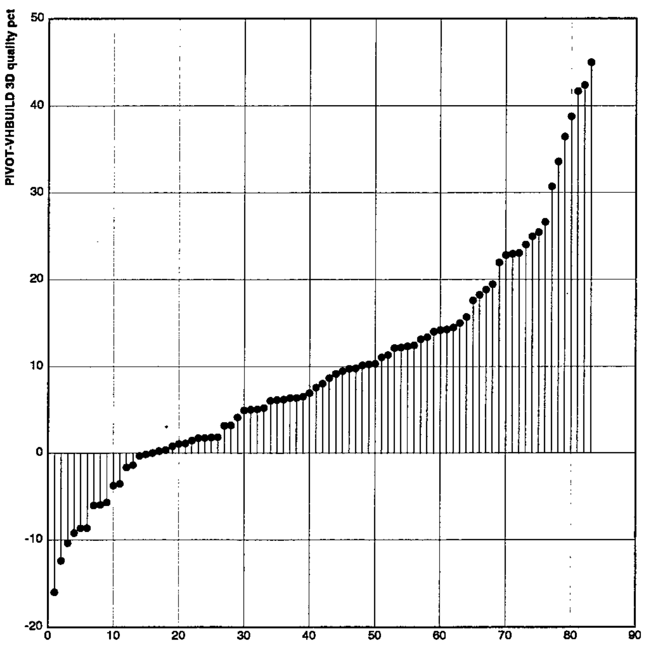


Figure 6-15: Relative 3D quality, VHBUILD vs PIVOT

In the next series of results, the performance of PIVOT is compared with that of BUILD+SHAVE. This comparison is particularly interesting because it contrasts two opposites of the building extraction spectrum. While there are many differences in the low-level and high-level algorithms which PIVOT and BUILD+SHAVE employ, the key difference between the two systems is that BUILD+SHAVE makes no use of a camera model, relying on nadir-based approximations of building shapes and shadow geometry, while PIVOT was designed and implemented on the supposition that rigorous camera modeling is a crucial aspect of object detection and delineation.

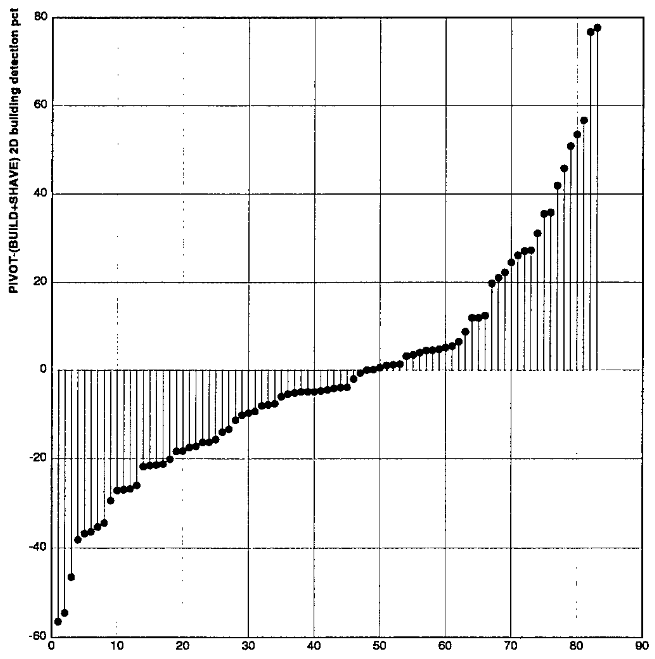


Figure 6-16: Relative 2D building detection percentage, BUILD+SHAVE vs PIVOT

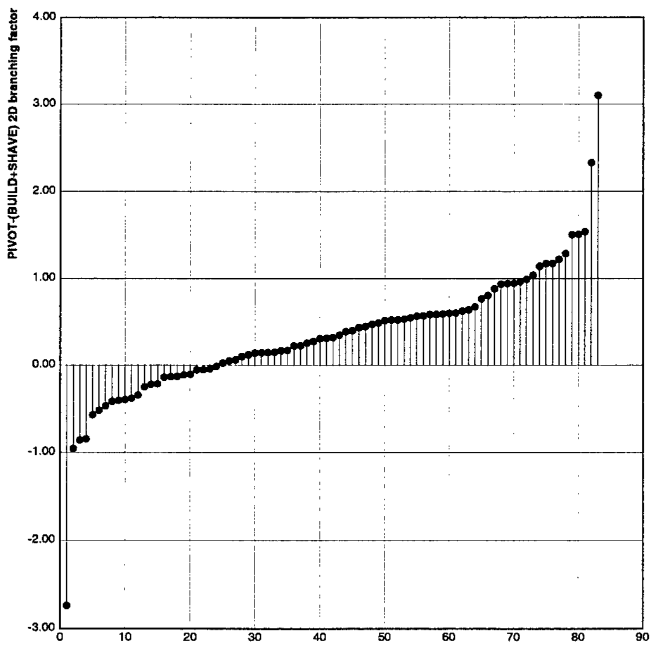


Figure 6-17: Relative 2D branching factor, BUILD+SHAVE vs PIVOT

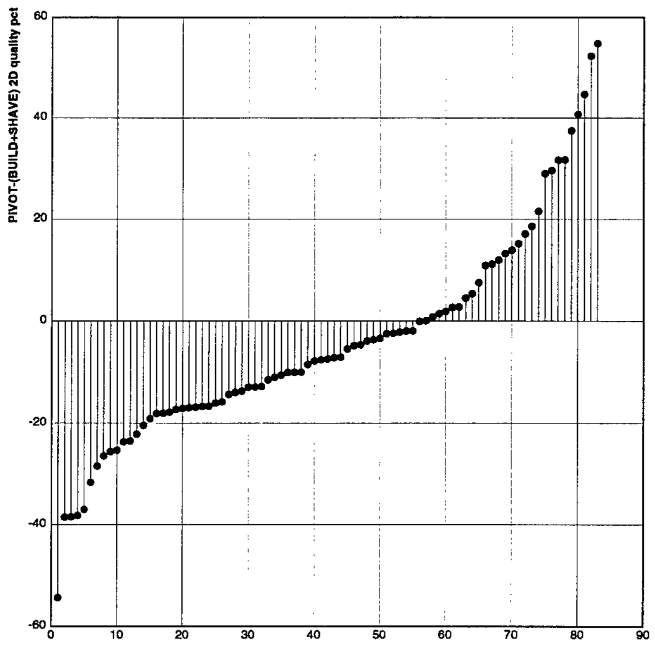


Figure 6-18: Relative 2D quality percentage, BUILD+SHAVE vs PIVOT

First, the 2D comparative statistics are presented in Figures 6-16 through 6-17. Figure 6-16 shows the relative differences in building detection percentages, subtracting BUILD+SHAVE's percentage from PIVOT's percentage. BUILD+SHAVE exhibits a better 2D building detection rate in 47 of the 83 scenes; most of this majority is represented by detection rate improvements of less than 20%. BUILD+SHAVE's advantage here can at least partially be attributed to its ability to generate building hypotheses from only one corner (two edges); PIVOT requires a 2-corner (three edges), which makes it more sensitive to low-level edge fragmentation problems or missing edges.

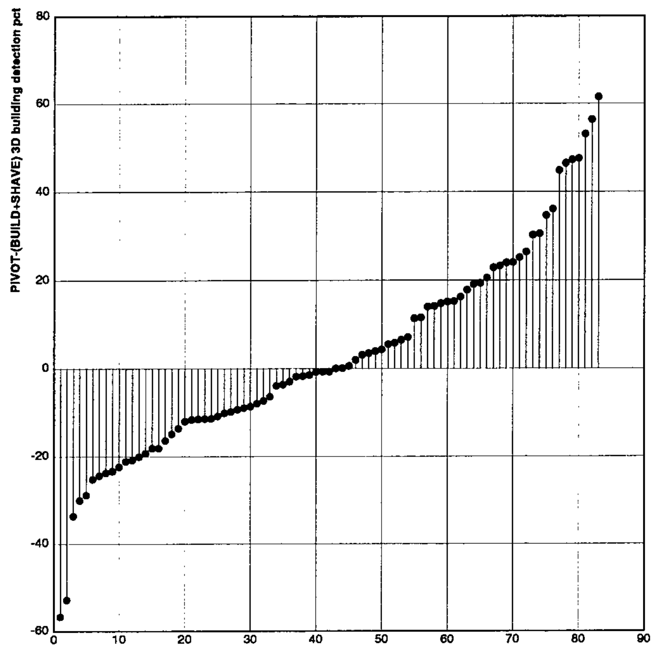


Figure 6-19: Relative 3D building detection percentage, BUILD+SHAVE VS PIVOT

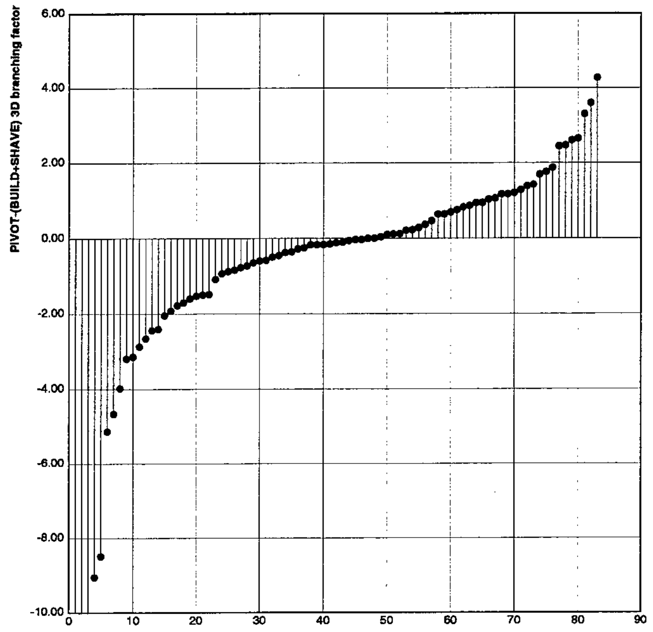


Figure 6-20: Relative 3D branching factor, BUILD+SHAVE VS PIVOT

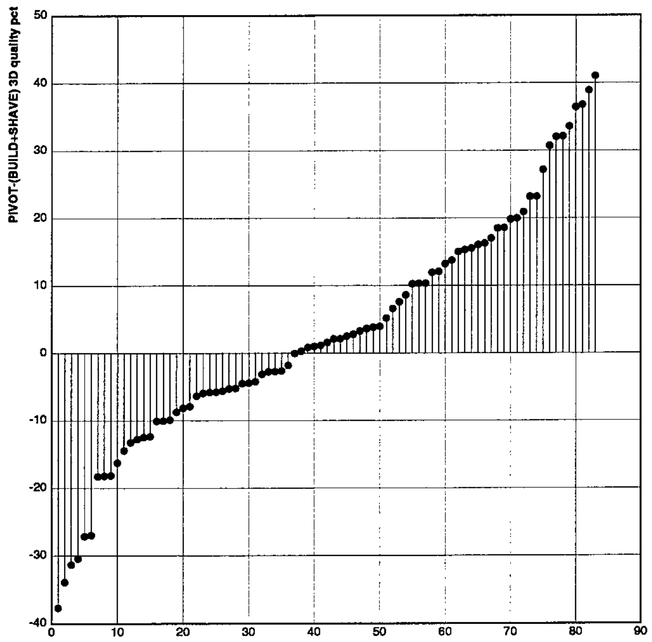


Figure 6-21: Relative 3D quality percentage, BUILD+SHAVE VS PIVOT

The 2D branching factor statistics, presented in Figure 6-17, show that in 59 of the test cases, PIVOT generates more false positives than BUILD+SHAVE. Combining this information with the slight overall superiority of BUILD+SHAVE in detecting building pixels, we expect the quality percentage metric to show that BUILD+SHAVE exhibits superior performance. As Figure 6-18 shows, this is indeed the case, with BUILD+SHAVE producing a higher quality percentage in 56 of the scenes. These statistics indicate that PIVOT's 2D hypothesis generation mechanisms are more sensitive than those used by

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**67**	17	-	-	-
BUILD+SHAVE	**41**	6	**38**	36	30	32
VHBUILD	10	7	4	8	16	10
PIVOT	32	9	24	**39**	**37**	**41**

Table 6-2: Relative performance summary

BUILD+SHAVE. Determining the nature of that sensitivity is deferred to Section 6.5; first, the 3D comparative statistics for BUILD+SHAVE and PIVOT are presented.

Figures 6-19 through 6-21 present the 3D relative performance statistics. Comparing all three figures with their 2D counterparts, a shift to the left can be observed, indicating that PIVOT's 3D performance is in some cases robust enough to alleviate errors in 2D detection and delineation. In particular, the relative performance statistics show no clear edge for either system, with PIVOT achieving better 3D building detection performance in 41 of the test scenes, better 3D branching factors in 47 of the scenes, and better 3D quality percentages in 46 of the scenes.

Table 6-2 provides an abbreviated summary of the performance statistics presented thus far. Each entry in the table, indexed by a system and a performance metric, indicates the number of images for which that system had the best value of that particular performance metric. Boldface numbers correspond to the systems exhibiting the best performance for a particular metric on a majority of the test scenes. Each column of the table sums to 83 (the 2D branching factor column is the only exception, as BUILD and BUILD+SHAVE had 6 ties for the best branching factor). At this stage, some general conclusions can be made about the performance of the building extraction systems:

- PIVOT exhibits modest improvements over VHBUILD in 2D object detection and delineation; in 3D, its performance improvements over VHBUILD are substantial, due to more robust 3D hypothesis generation mechanisms, enabled by improved vanishing point modeling and height estimation methods.
- BUILD and BUILD+SHAVE exhibit comparable performance in 2D, where BUILD+SHAVE's increased branching factor is moderated by its increased building detection rate.
- PIVOT and BUILD+SHAVE exhibit comparable performance in 3D, where PIVOT's relatively worse 2D detection and delineation performance is obviated by its 3D hypothesis generation mechanisms.

These conclusions are necessarily general, however. While the performance statistics used here provide an unbiased picture of overall performance, they do not give clear insight into specific performance strengths and weaknesses. For example, in Figure 6-21, the quality percentages of PIVOT and VHBUILD differ by at most 10% on a majority of the scenes. However, for some scenes, PIVOT significantly outperforms BUILD+SHAVE, in one case by over 40%. Conversely, BUILD+SHAVE outperforms PIVOT significantly for several scenes. It should be clear that certain images present special difficulties for each of these systems, or, to be more explicit, for the particular hypothesis generation and verification methods used in each system. As noted earlier, PIVOT and BUILD+SHAVE use essentially different methods at every phase of the building extraction process, from edge generation to corner finding to 2D structure generation and analysis to hypothesis verification to 3D object model generation. Understanding the features of images and scenes which present particular difficulties for these methods forms the central topic of the next section.

6.5. Image/scene complexity and its impact on performance

The performance analysis in Section 6.4 suggested that some images and scenes present particular difficulties for various feature extraction and hypothesis generation algorithms. In particular, the performance comparisons between PIVOT and BUILD+SHAVE showed that PIVOT was able to perform very well on some scenes where BUILD+SHAVE performed quite poorly, and vice versa. As noted earlier in Section 6.3, understanding why particular algorithms have robust behavior on certain kinds of imagery while failing on others leads gives insights for improvements of those algorithms. This leads naturally to two important questions: which characteristics of the image and/or scene affect algorithm performance, and why?

To answer these questions, we seek system-independent measures of *image complexity* and *scene complexity*, in terms which can be directly related to the underlying algorithms in object detection and delineation systems. System-independence is an important property of such measures, in that it allows comparisons to be made across algorithms. One could try to define the complexity of an image to simply be the inverse of the 3D quality percentage produced by PIVOT, but this is clearly insufficient as it gives no insights into the properties of images which cause performance failures in PIVOT. For example, if PIVOT performs poorly on some large varied set of images, and BUILD+SHAVE performs well on those images, one might reach the conclusion that BUILD+SHAVE performs well on scenes of high complexity. While this conclusion would be true, it would also be vacuous, since none of the properties which led to PIVOT's performance breakdowns would have been identified. Given the results in the previous section, the situation would be even worse, as there would be no correlation between image complexity and BUILD+SHAVE's performance.

There are several properties of images and scenes which can be expected to affect the performance of 3D object detection and delineation algorithms. Among these are the obliquity of the viewing angle, the degree of low-level edge fragmentation, the density of edges, the complexity of object shapes, the density of object shapes, image scale, visibility of shadows, the degree of image contrast along physical object boundaries, the extent and position of occluding objects, the dynamic intensity range of the image, the relative intensity of objects with respect to their shadows and the background, and the number of object orientations, among others.

Many of these factors are interdependent; for example, the degree of edge fragmentation is correlated with edge contrast, since less contrast across object boundaries usually translates to low response from edge operators. Similarly, shadow visibility is a function of viewing angle, object position, size, and shape. Given this interdependence, it is unlikely that any single "image/scene complexity measure" can usefully quantify all of the factors which affect system performance. A better approach is to look at specific factors relative to the performance metrics, and examine system behavior with respect to these factors, with the intent of correlating performance extrema with specific image/scene properties.

A useful way to view image and scene complexity analyses is in terms of a metric space, where each dimension of the space corresponds to a particular measurable image or scene property. As the experimental results in Section 6.4 show, each of the building extraction systems performs with varying degrees of success in different parts of this *image/scene complexity space*. This formulation suggests the following questions:

- For a given portion of the space, which of the object detection and delineation systems exhibits the best performance?
- In which portions of the space does a given system exhibit good performance and poor performance in absolute terms?
- How does a system's performance correlate with various dimensions of the image/scene complexity space?

Of course, these questions can only be answered to the extent that the test imagery supports such analysis. While the 83 images used here represent a significant test, particularly with respect to much of the previous work in building extraction (see Table 1-4 on page 7), they cover only a small fraction of the image/scene complexity space. For example, the average scale of the images used in this work ranges from 0.17*m* to 0.90*m* per pixel; the average scale of 73 of these images lies between 0.20*m* and 0.60*m* per pixel. Images taken by satellite multispectral sensor platforms such as SPOT often have resolutions on the order of 30*m* per pixel; the other resolution extreme is represented by images acquired for architectural photogrammetry and other close-range applications, where images are expected to support measurement accuracies on the order of 1*mm* or less [94]. So, although it is reasonable to expect performance breakdowns as scale deviates from the 0.2*m* to 0.9*m* range, no conclusions can be drawn in the absence of test imagery outside of this range, and any conclusions that are made can only be expected to hold true for images with scales within this range.

Nonetheless, the test imagery does support analyses of the effects of certain image and scene properties which are well represented. The following sections present analyses and case studies designed to examine individual properties of the images and scenes, with respect to the building detection percentage, branching factor, and quality percentage performance metrics. In this section, two sets of experiments examine the effects of image obliquity and object complexity/density on performance over the 83 test scenes.

6.5.1. Effects of image obliquity

Many building extraction systems, unlike PIVOT, are based around the assumption that images are taken from a vertical viewpoint. BUILD and BUILD+SHAVE both use this assumption in locating corners; if the image is vertical, then horizontal right-angled corners in the scene appear as right-angled corners in the image, which simplifies the corner finding task. BUILD+SHAVE uses the vertical viewpoint assumption for height mensuration of buildings, and BUILD, BUILD+SHAVE, and VHBUILD all use the same assumption in computing shadow regions for hypothesis verification.

As a consequence, it should generally be expected that the performance of these systems should degrade as the viewing angle deviates from vertical. There are exceptions; if the deviation from vertical lies along one of the dominant horizontal orientations of buildings in the scene, then the right angles of the rooftops of these buildings will still appear as right angles in the image, and systems such as BUILD+SHAVE will still generate acceptable roof polygons despite the image obliquity. However, the imaged shadow geometry also changes with viewpoint, and it should be expected that BUILD+SHAVE's ability to generate reasonable height estimates for structures will degrade accordingly. In this section, the effects of changes in viewing angle on building extraction performance are evaluated on the 83 test scenes.

The *image tilt* angle is defined as the angle formed by the optical axis with respect to the *Z*-axis of the world coordinate system; Equation (A.6) in Appendix A gives the formula for image tilt. In photogrammetric terms, a *vertical* photograph is considered to be one which has a small tilt angle, typically on the order of 3 degrees; a *low oblique* photograph has more tilt than a vertical photograph; and a *high oblique* photograph has enough tilt for the apparent horizon to appear in the field of view [94]. For the purposes of evaluation in this work, a vertical photograph will be defined as one with less than 5 degrees of tilt, a low oblique photograph will be defined to have between 5 and 30 degrees of tilt, and high oblique photos will have greater than 30 degrees of tilt.

Figure 6-22 plots the image tilt angles (in degrees) for the 83 test scenes, sorted by increasing angle from left to right. The triple lines at 5 degrees and 30 degrees indicate the boundaries between vertical, low oblique, and high oblique imagery. Here, note that half of the test images come from vertical frame photography, and the other half comes from oblique photography.

However, one difficulty with the use of image tilt as a measure of obliquity is that it is most accurate near the principal point of the frame. For images which are extracted from the sides of a photograph, rather than the center, the image tilt for the entire frame becomes a less accurate approximation of the obliquity in the image patch. To get a more representative measure of obliquity for the test images, an alternate measure is used. The *effective obliquity* angle of an image is defined as the angle formed by a ray passing through the perspective center and the center of the image with respect to the *Z*-axis of the world coordinate system.

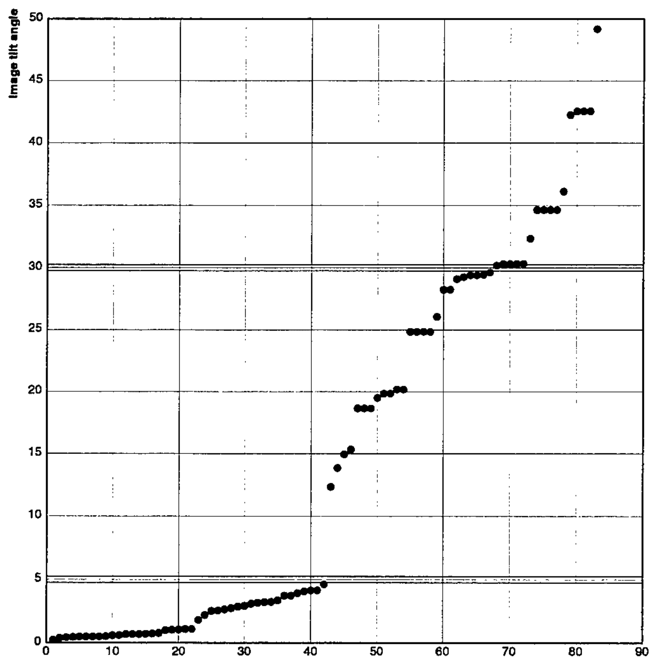


Figure 6-22: Image tilt angles for test scenes

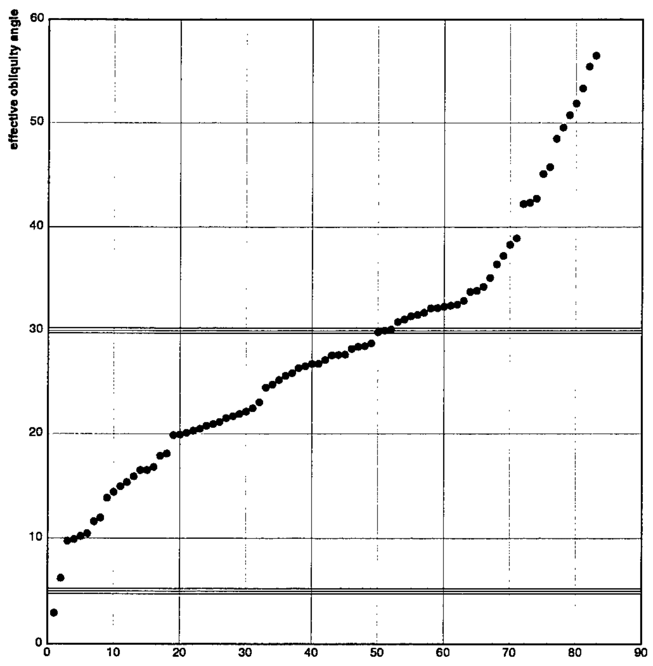


Figure 6-23: Effective obliquity for test scenes

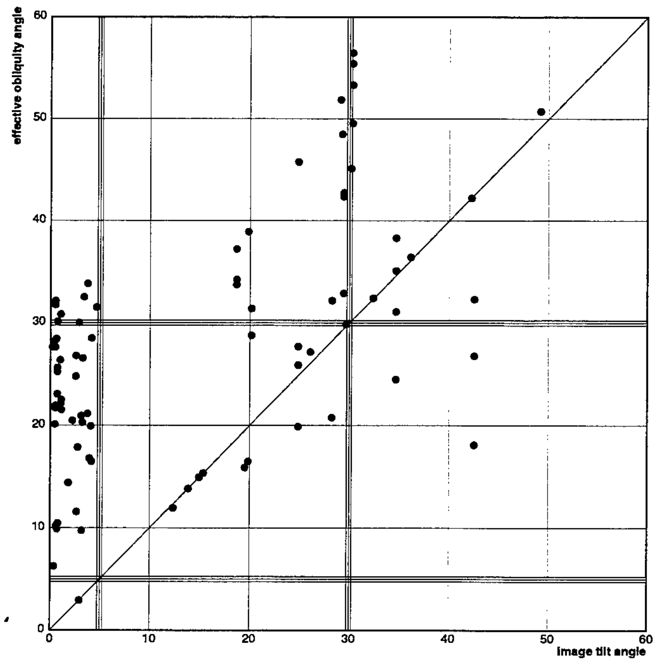


Figure 6-24: Comparison of image tilt vs effective obliquity

Figure 6-23 plots the effective obliquity angles (in degrees) for the 83 test scenes, sorted by increasing angle from left to right, with the same sets of triple lines as in Figure 6-22 to denote the boundaries between vertical, low oblique, and high oblique angles. Here, observe that only one image has a vertical effective obliquity; 49 images are low oblique, and 33 are high oblique.

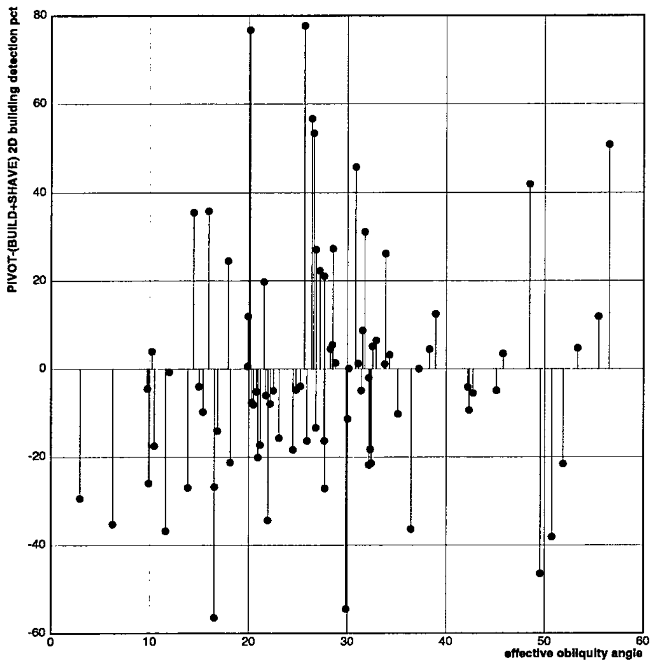


Figure 6-25: Relative 2D bld detection vs effective obliquity, BUILD+SHAVE vs PIVOT

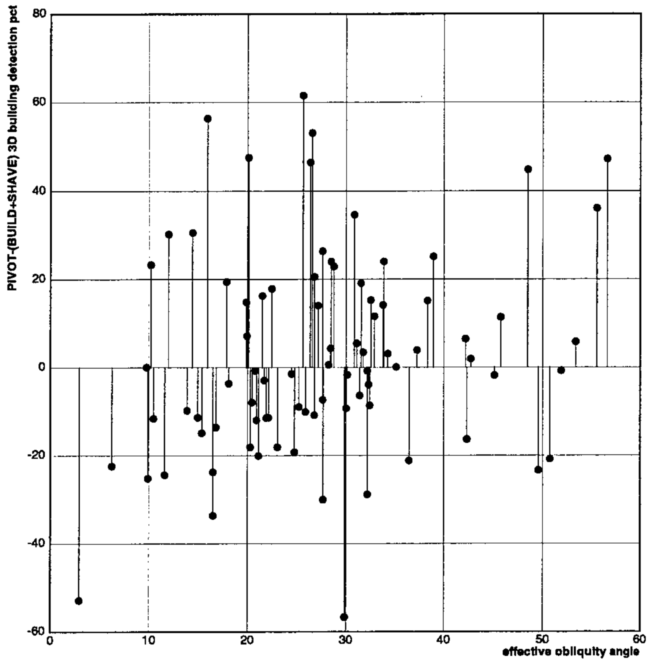


Figure 6-26: Relative 3D bld detection vs effective obliquity, BUILD+SHAVE vs PIVOT

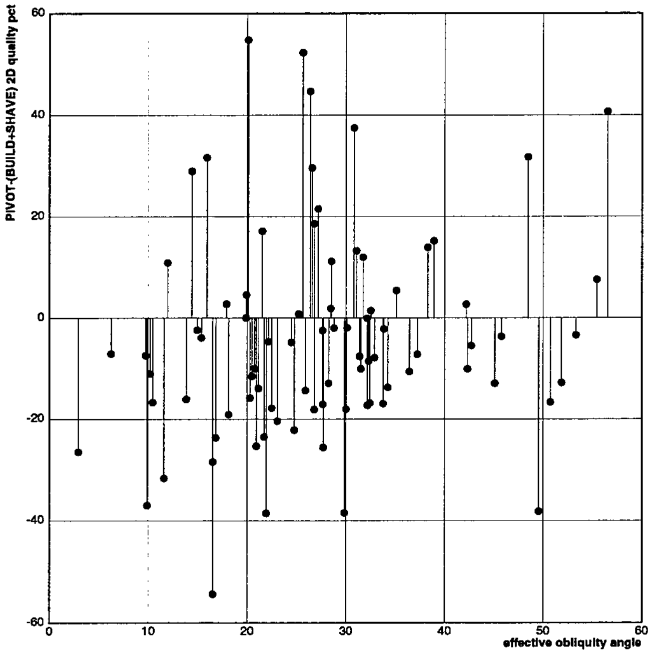


Figure 6-27: Relative 2D quality pct vs effective obliquity, BUILD+SHAVE vs PIVOT

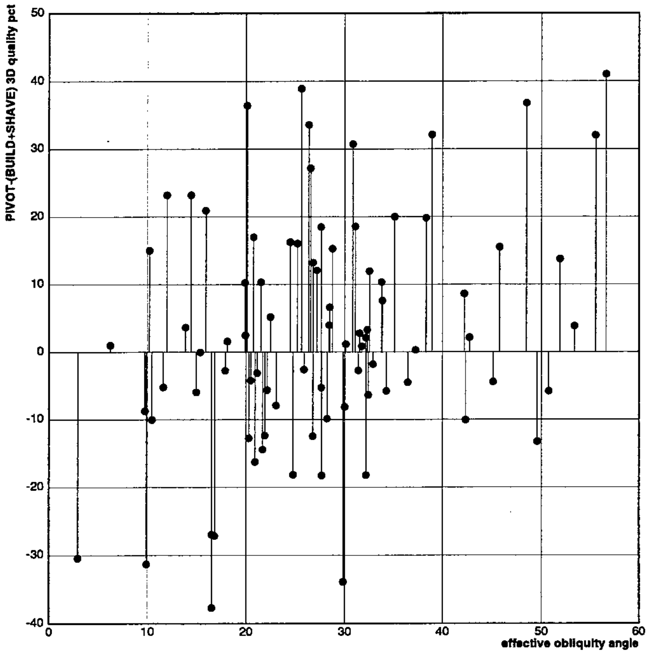


Figure 6-28: Relative 3D quality pct vs effective obliquity, BUILD+SHAVE vs PIVOT

Finally, Figure 6-24 plots the image tilt angle against the effective obliquity angle for each image. Note that many of the images classified as vertical by the image tilt angle actually have significant obliquity, some even high enough to be classified as high oblique. Also note that only a few points lie on or near the line, meaning only a few of the test images were actually extracted near the principal point of the image. This illustrates that image tilt is only a reasonable approximation of obliquity near the principal point of the image, which is to be expected for typical aerial mapping photography.

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**1**	0	-	-	-
BUILD+SHAVE	**1**	0	**1**	**1**	0	**1**
VHBUILD	0	0	0	0	0	0
PIVOT	0	0	0	0	**1**	0

Table 6-3: Relative performance summary, vertical (nadir) imagery (effective obliquity)

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**42**	11	-	-	-
BUILD+SHAVE	**28**	5	**22**	**25**	**22**	21
VHBUILD	6	2	3	5	7	6
PIVOT	15	5	13	19	20	**22**

Table 6-4: Relative performance summary, low oblique imagery (effective obliquity)

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**24**	6	-	-	-
BUILD+SHAVE	12	1	**15**	10	8	10
VHBUILD	4	5	1	3	9	4
PIVOT	**17**	4	11	**20**	**16**	**19**

Table 6-5: Relative performance summary, high oblique imagery (effective obliquity)

It should be noted in passing that scenes with high effective obliquity angles taken from nearly vertical images will not exhibit geometric effects sufficiently significant to affect the performance of nadir-based building extraction systems; right-angled horizontal line pairs in the world will image as right angles, since the horizontal vanishing points are at infinity. However, the geometry of vertical line segments will still vary across the image, and the effective obliquity angle captures this effect, which can be crucial for systems like VHBUILD which rely on verticals for height mensuration of buildings. For comparison purposes, the effective obliquity angle will be used as the measure of obliquity.

Figures 6-25 through 6-28 show the relative change in performance from BUILD+SHAVE to PIVOT, for the building detection rate and quality percentage (the branching factor is omitted from subsequent analyses for brevity, as the effects of false positives are factored into the quality percentage).

The most prominent patterns are to be seen in the transition from 2D to 3D. For the 2D and 3D relative building detection percentages plotted in Figures 6-25 and 6-26, respectively, note the general upward shift of points, reflecting the performance improvement due to PIVOT's more rigorous implementation of 3D geometry. Given the overall statistics in Section 6.4, this is to be expected, but note that PIVOT appears to achieve noticeably better 3D building detection performance for high oblique imagery (effective obliquities greater than 30 degrees) than for low oblique imagery (effective obliquities between 5 and 30 degrees).

The trend stands out more sharply in the relative quality percentage plots in Figures 6-27 and 6-28, where BUILD+SHAVE's simplistic model of shadow geometry leads to greater inaccuracies in height mensuration and a corresponding increase in false positive voxels. The relative upward shift from 2D to 3D is clearly evident in these plots, with PIVOT again achieving a significant performance improvement over BUILD+SHAVE on the high oblique images.

Tables 6-3 through 6-5 provide an abbreviated summary of the performance statistics with respect to the three obliquity classes. As with the overall performance statistics presented earlier in Table 6-2, each entry in a table indicates the number of images for which a particular system achieved the best value for a particular metric. Table 6-3 is included for completeness, but no conclusions can be drawn from these results, as only one image fell into the vertical category.

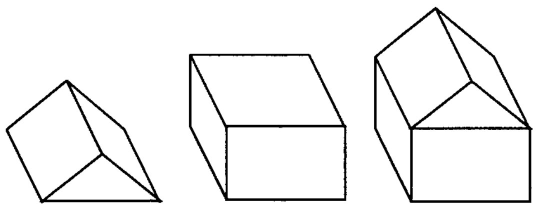
For the 49 low oblique images, summarized in Table 6-4, BUILD+SHAVE enjoys a clear performance superiority in terms of 2D building detection and 2D quality percentage, while the 3D performance statistics show that both BUILD+SHAVE and PIVOT split the majority of images relatively evenly. It should be noted that the 2D branching factor statistics reflect 5 ties between BUILD and BUILD+SHAVE, and so that column does not sum to 49. It should also be noted that BUILD's large majority in the 2D branching factor primarily reflects BUILD's generally lower number of false positives due to its generation of rooftop polygons only, and not wall polygons.

For the 33 high oblique images, summarized in Table 6-5, PIVOT's performance in 2D building detection and quality relative to the competing systems is improved over its performance on the low oblique images. In 3D building detection and quality, PIVOT achieves the best performance in all three performance metrics by a significant margin. Again, it should be noted that 1 tie between BUILD and BUILD+SHAVE is reflected in the 2D branching factor statistics.

The results presented in this section show that on average, PIVOT achieves superior 3D performance to BUILD+SHAVE as the effective obliquity of the imagery increases from low oblique to high oblique. This is consistent with the expectation that PIVOT's 3D object space modeling of shadow geometry and object geometry should lead to better building detection and delineation performance than its nadir-based counterparts. The 2D statistics also show an improvement for PIVOT as the obliquity increases from low to high, although here BUILD+SHAVE is more likely to outperform PIVOT, due to its ability to generate building hypotheses from fewer edges, as mentioned earlier. Overall, these results support the hypothesis that rigorous modeling of 3D geometry enables improvements in building detection and delineation performance, particularly over a wide range of viewpoints.

6.5.2. Effects of object complexity and density

Building detection and delineation systems generate scene models in a variety of representations, some of which allow for more complex building structures than others. For example BUILD+SHAVE only models buildings as individual 3D quadrilateral volumes, while PIVOT models buildings as compositions of rectangular and triangular volumes. As a consequence, it is to be expected that PIVOT should exhibit superior performance in modeling buildings with greater shape complexity than BUILD+SHAVE. A measure of *object complexity* serves to quantify the representational demands of a scene on an object detection and delineation system.



v	6	8	10
e	9	12	17
f	5	6	9
c_{OBJ}	20	26	36

Figure 6-29: Examples of object complexity

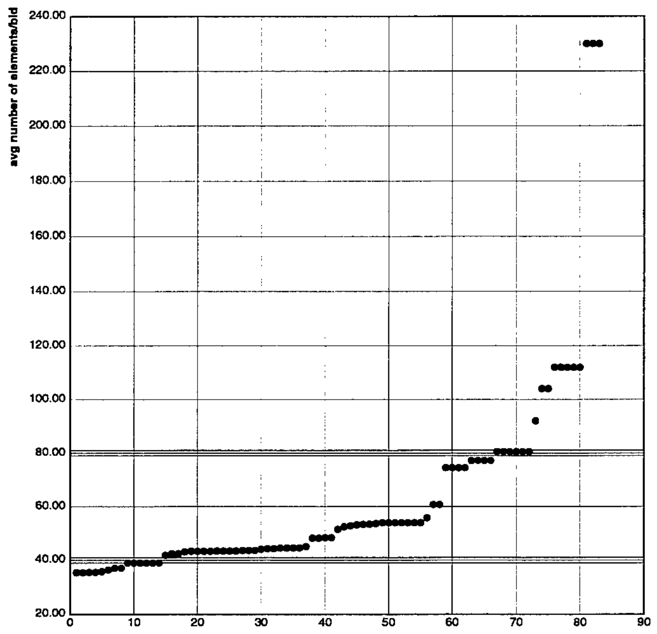


Figure 6-30: Avg object complexity for test scenes

There are several ways to quantify the complexity c_{OBJ} of an object. In this work, a simple measure is used, assuming that objects are represented as wireframe models:

$$c_{OBJ} = v + e + f \qquad (6.1)$$

where v is the number of vertices of the object, e is the number of edges, and f is the number of faces; c_{OBJ} will often be referred to as the number of *elements* of an object. There is a degree of redundancy in this complexity measure, since by Euler's formula, $v+f = e+2$ for simple polyhedra. It is also too simplistic a measure for curved surfaces, such as spheres and cylinders, which require a large number of polygons to be approximated with high fidelity, but which have trivial parametric representations. Nonetheless, for this task domain, where the vast majority of structures are best modeled as wireframes, it is a reasonable measure of object complexity. Figure 6-29 shows the object complexity measure for the triangular prism, the rectangular volume, and a peaked/gabled roof building model.

Figure 6-30 shows the average number of elements for the ground truth building models in each test scene, sorted in increasing order from left to right. The triple bars are used to denote regions of low, moderate, and high object complexity, separated at 40 and 80 elements per building; the three outlying images with the highest average complexity correspond to the MMBLD scene, which consists of one multi-level rectilinear structure with several corners and walls on each level (see Figures B-137, B-141, and B-145 in Appendix B). The bulk of the images lie in the moderate object complexity class, which is typically the result of a mixture of several low complexity buildings and a few high complexity buildings. For example, the J scene has 7 images which fall into this category (Figure B-97 shows one of the views of this area).

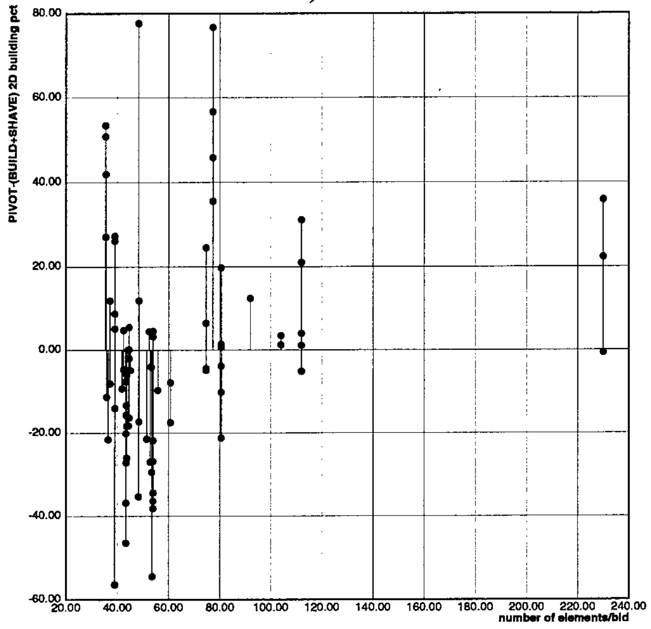


Figure 6-31: Relative 2D bld detection vs object complexity, BUILD+SHAVE vs PIVOT

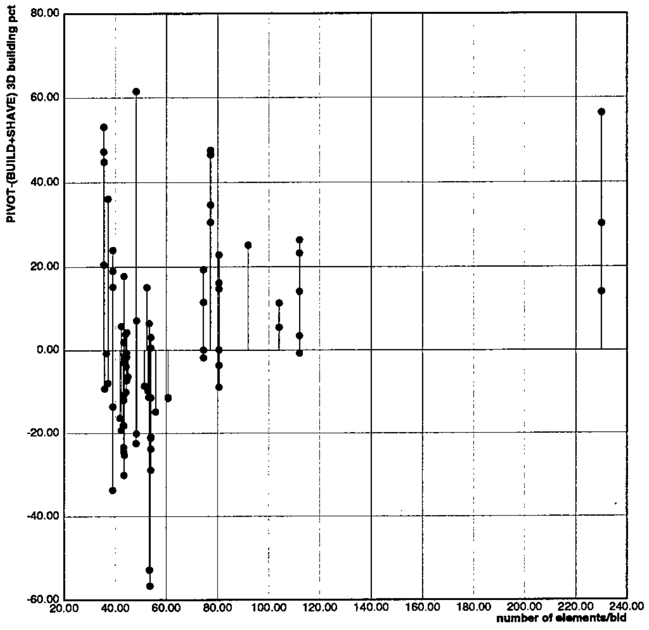


Figure 6-32: Relative 3D bld detection vs object complexity, BUILD+SHAVE vs PIVOT

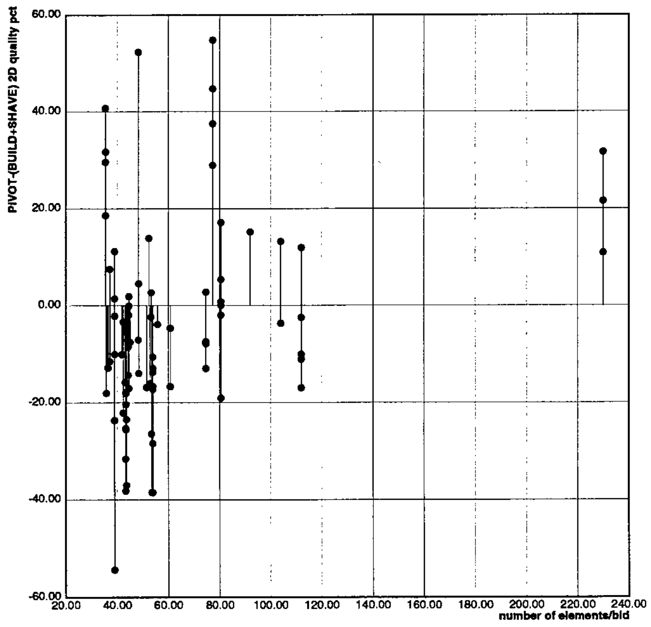


Figure 6-33: Relative 2D quality pct vs object complexity, BUILD+SHAVE vs PIVOT

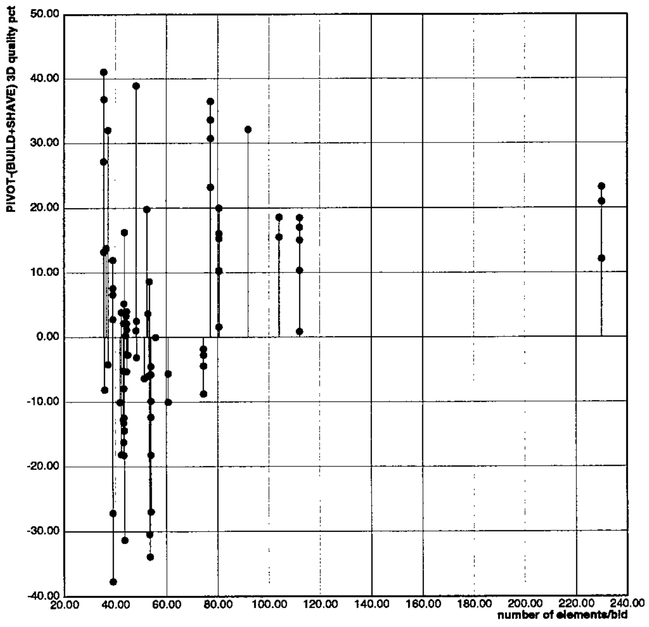


Figure 6-34: Relative 3D quality pct vs object complexity, BUILD+SHAVE vs PIVOT

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**11**	2	-	-	-
BUILD+SHAVE	3	1	4	4	**5**	4
VHBUILD	4	2	2	2	4	1
PIVOT	**7**	1	**6**	**8**	**5**	**9**

Table 6-6: Relative performance summary, low object complexity

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**45**	10	-	-	-
BUILD+SHAVE	**34**	5	**31**	**30**	**24**	**28**
VHBUILD	3	2	1	4	6	8
PIVOT	15	5	10	18	22	16

Table 6-7: Relative performance summary, moderate object complexity

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**11**	5	-	-	-
BUILD+SHAVE	4	0	3	2	1	0
VHBUILD	3	3	1	2	6	1
PIVOT	**10**	3	**8**	**13**	**10**	**16**

Table 6-8: Relative performance summary, high object complexity

As in the previous analysis, Figures 6-31 through 6-34 give the relative building detection and quality percentage statistics in both 2D and 3D, compared against the average object complexity for each scene.

In absolute terms, two features stand out in all four figures. First, PIVOT has a noticeable performance edge on images with high object complexity (where the average number of elements per building is greater than 80). This is due to PIVOT's ability to model more complex buildings as collections of primitives. On the other hand, BUILD+SHAVE shows a noticeable performance edge on images with moderate object complexity (where the average number of elements per building is between 40 and 80).

Tables 6-6 through 6-8 provide the abbreviated summaries of performance statistics with respect to the three object complexity classes. As before, each entry in a table indicates the number of images for which a particular system achieved the best value for a particular metric. Also, as in previous tables, there are 6 ties for best 2D branching factor between BUILD and BUILD+SHAVE, so those columns do not always sum to 83 test images.

Tables 6-6 and 6-7 show that BUILD+SHAVE performs better for images with low object complexity, with a significant discrepancy in performance for images with moderate object complexity, in terms of both 2D and 3D building detection percentage and quality percentage. The 2D and 3D branching factor statistics, however, are evenly split for both low and moderate object complexity, which indicates that the bulk of the discrepancy is due to better detection performance on the part of BUILD+SHAVE. This improved detection performance is hard to attribute to any specific cause, but one probable factor is BUILD+SHAVE's ability to generate complete building models from an initial set of two edges, while PIVOT requires three. In cases where poor edge gradient and edge fragmentation are frequent, BUILD+SHAVE has a better chance of detecting a building as it requires less data. The downside of this approach is that it also leads to more frequent generation of false positives, which is consistent with the more evenly matched branching factor statistics.

Table 6-8 shows the summary of performance statistics for images with high object complexity. For these images, PIVOT achieves the best overall performance in building detection and quality percentage for both 2D and 3D, with the best 3D quality percentage in 16 out of the 17 high complexity images. This is consistent with the expectation that PIVOT's more general object representation scheme is better suited to handle complex buildings.

While average object complexity is one measure of a scene's complexity, it ignores building frequency. As buildings become increasingly dense in a scene, the possibility for inter-building occlusions increases, as does the chance that edges and corners belonging to one building may be linked erroneously to those generated for another nearby building. In scenes where road networks and buildings lie on a perpendicular grid, as is often the case in urban areas, this latter problem becomes a significant detriment to performance. The expectation is that as the *element density* increases, systems which use local perceptual grouping techniques to group supposedly related features will increasingly exhibit detection and delineation breakdowns, as unrelated features are linked together. Principle 4 argued against such local strategies for intermediate feature generation.

Element density can be measured in two complementary ways, in object space and image space. By measuring the average number of elements per unit area in object space, the *physical* density of points, edges, and surfaces in the scene is obtained; by measuring the average number of elements per unit area in image space, the *imaged* density of points, edges, and surfaces in the scene is obtained. Both measurements are important to obtain a clear picture of performance effects with respect to element density. For example, buildings which are physically far apart may appear quite close in terms of pixel distance in the image, if image resolution is sufficiently low, or if image obliquity is sufficiently high. Loosely speaking, for a given scene density d, the image density is proportional to $d/\cos\tau$, where τ is the image tilt angle (defined by Equation (A.6) in Appendix A). As the tilt angle increases, the imaged density also increases (as well as the number of inter-element occlusions). By measuring element density in image space (2D) and in object space (3D), performance trends in either space can be evaluated.

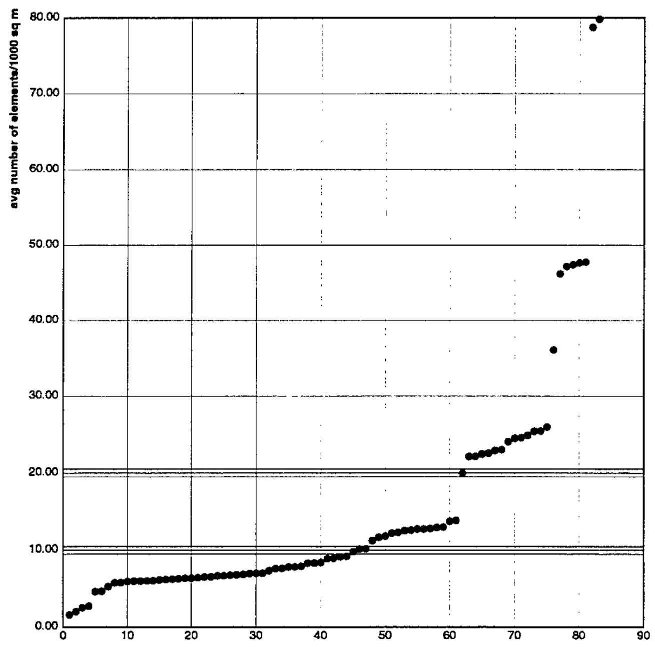


Figure 6-35: Object space density of elements

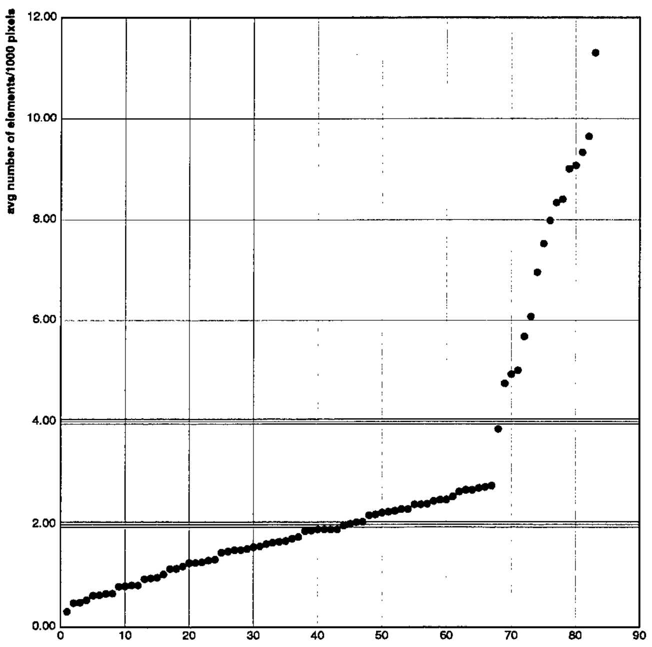


Figure 6-36: Image space density of elements

Figure 6-35 shows the object space density of elements in the scene, computed by counting the total number of elements in the ground truth building models and dividing by the ground surface area covered by the image. The densities are sorted in increasing order from left to right. Two sets of triple bars denote regions of low, moderate, and high density, separated at 10 and 20 elements per $1000m^2$. 45 of the images have low object space density, 17 have moderate density, and 21 have high density. The SLAG, VANILLA, and NEST images make up most of the high density images; the two highest density images correspond to the VILLAGE scene (see Figures B-325 and B-329 in Appendix B).

Figure 6-36 shows the image space density of elements in the scene, computed by dividing the total number of ground truth elements by the number of pixels in the image. Densities are once again sorted from left to right. Two sets of triple bars denote regions of low, moderate, and high density, separated at 2 and 4 elements per 1000 pixels.

Here, 44 images have low image space element density, 24 have moderate density, and 15 have high density. SLAG, VANILLA, and NEST images comprise most of the high density images again; NEST_FHOV1027 has the highest image density, which is understandable since it is has several buildings in an oblique view (see Figure B-157). The VILLAGE scenes have only moderate density in image space, due to their high resolution (less than $0.2m$ ground sample distance).

Figures 6-37 through 6-40 give the relative building detection and quality percentage statistics in both 2D and 3D, as a function of average object space element density, for each scene.

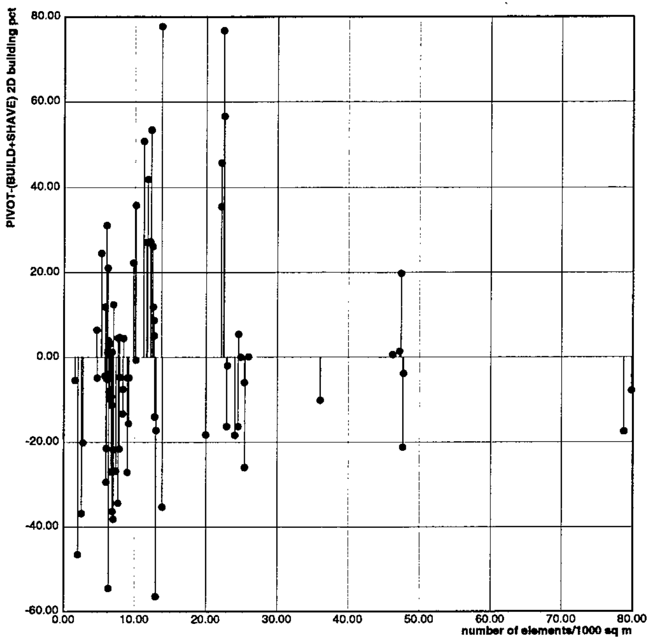


Figure 6-37: Relative 2D bld detection vs object space density, BUILD+SHAVE vs PIVOT

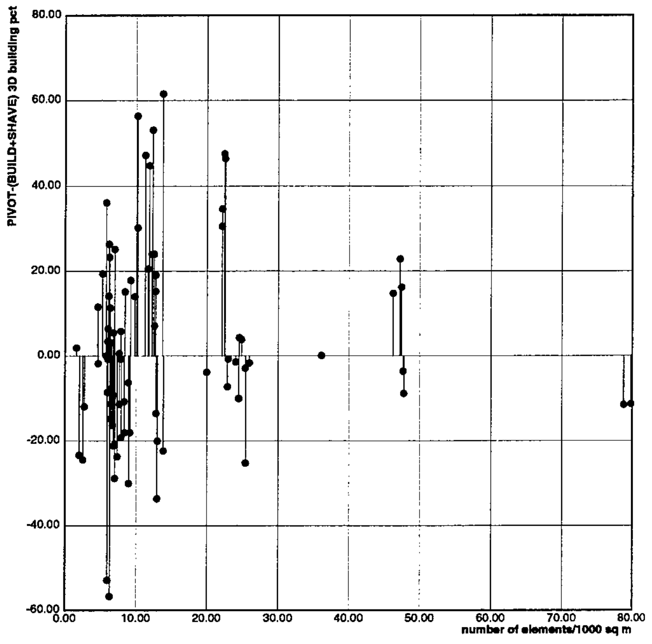


Figure 6-38: Relative 3D bld detection vs object space density, BUILD+SHAVE vs PIVOT

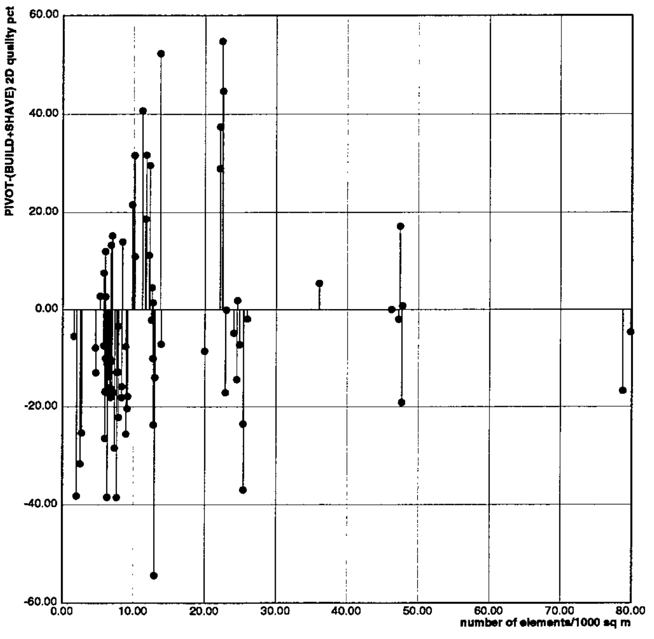


Figure 6-39: Relative 2D quality pct vs object space density, BUILD+SHAVE vs PIVOT

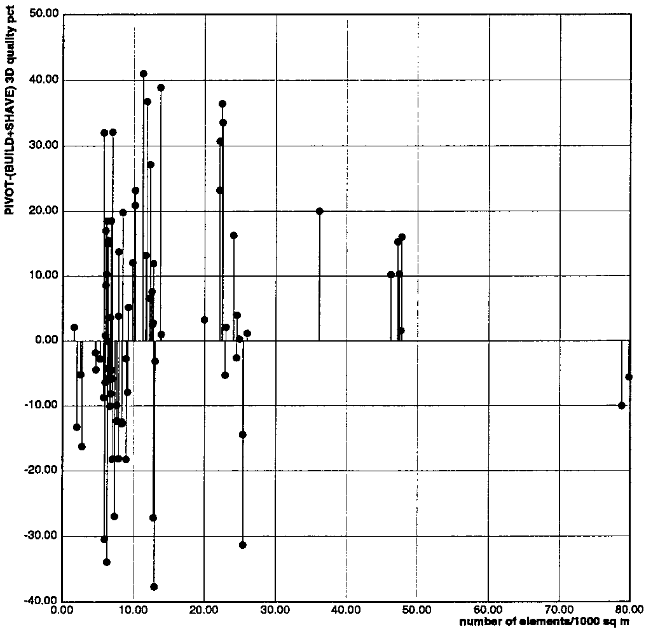


Figure 6-40: Relative 3D quality pct vs object space density, BUILD+SHAVE vs PIVOT

Three trends appear in each graph. First, a downward spike at the left of each graph indicates that BUILD+SHAVE exhibits superior performance in images of low object space element density. In all four graphs, this is immediately followed by an upward spike, indicating PIVOT's superior performance on images of moderate object space element density. Finally, the high density images (more than 20 elements per $1000m^2$) generally show a relative performance improvement in favor of PIVOT, although the difference is not as dramatic as in the moderate density images.

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**38**	9	-	-	-
BUILD+SHAVE	**26**	0	**27**	**23**	**19**	**24**
VHBUILD	5	3	2	4	7	6
PIVOT	14	4	7	18	**19**	15

Table 6-9: Relative performance summary, low object space element density

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**11**	3	-	-	-
BUILD+SHAVE	5	2	3	5	4	3
VHBUILD	3	3	2	1	6	2
PIVOT	**9**	3	**9**	**11**	**7**	**12**

Table 6-10: Relative performance summary, moderate object space element density

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**18**	5	-	-	-
BUILD+SHAVE	**10**	4	**8**	8	7	5
VHBUILD	2	1	0	3	3	2
PIVOT	9	2	**8**	**10**	**11**	**14**

Table 6-11: Relative performance summary, high object space element density

The performance summary statistics in Tables 6-9 through 6-11 also support these general trends. In Table 6-9, which summarizes the results for the low object space density images, BUILD+SHAVE shows better relative performance in 2D building detection and quality percentage, although the relative advantage is not as significant in 3D. For the moderate density images, summarized in Table 6-10, PIVOT consistently performs better on a majority of the images in both 2D and 3D building detection percentage and quality percentage. The relative performance improvements are not as clear for the high density images in terms of 2D performance statistics, presented in Table 6-11, but PIVOT does have a slight advantage in 3D building detection and branching factor, and a significant advantage in 3D quality percentage.

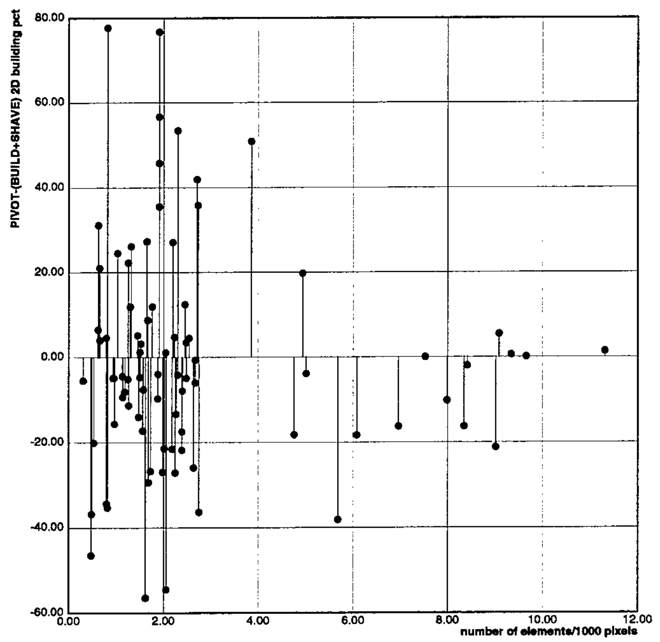


Figure 6-41: Relative 2D bld detection vs image space density, BUILD+SHAVE vs PIVOT

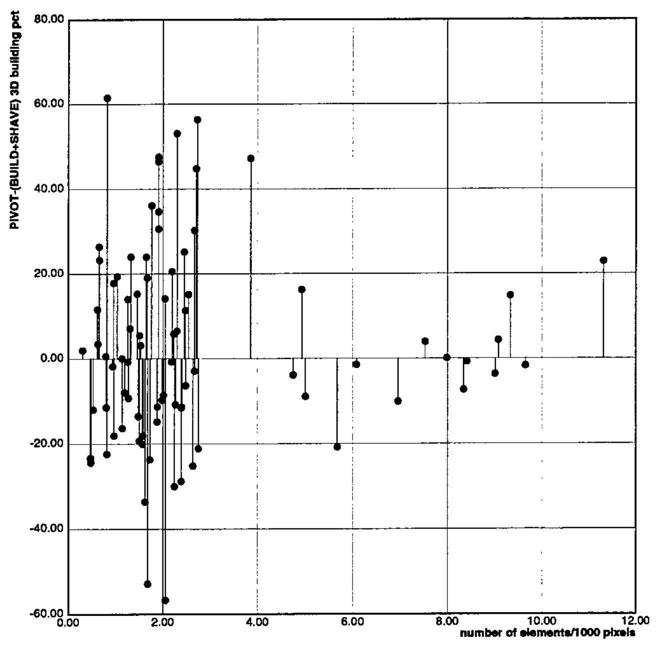


Figure 6-42: Relative 3D bld detection vs image space density, BUILD+SHAVE vs PIVOT

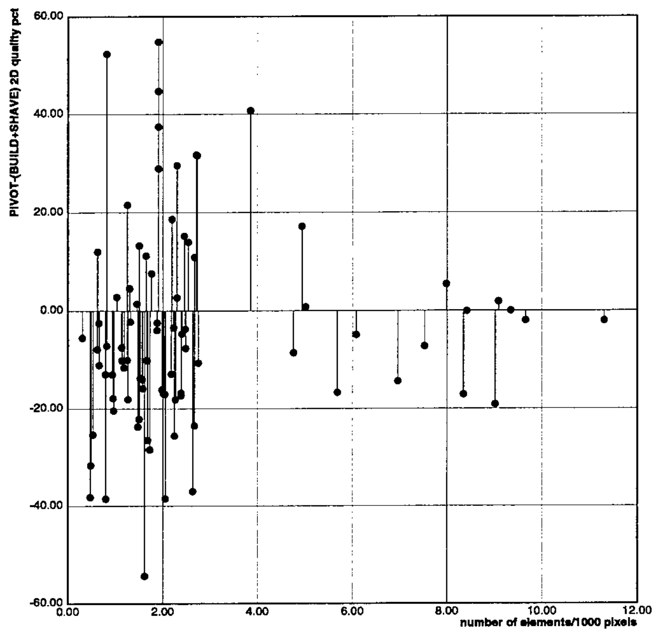


Figure 6-43: Relative 2D quality pct vs image space density, BUILD+SHAVE vs PIVOT

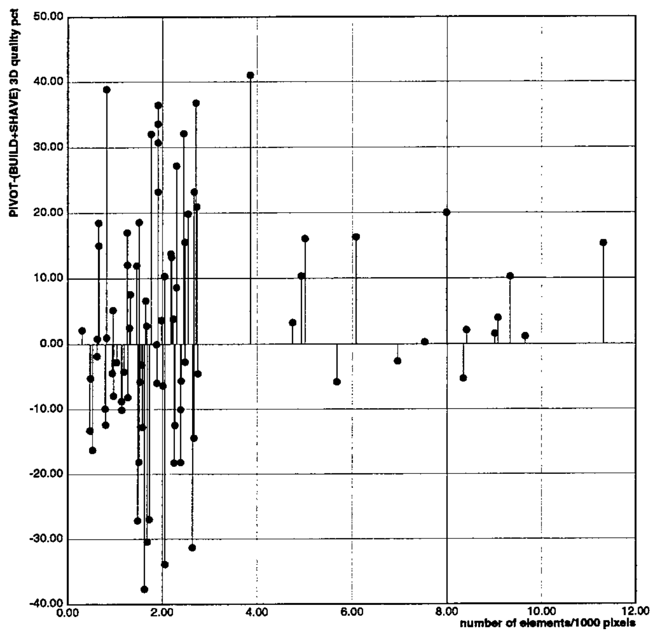


Figure 6-44: Relative 3D quality pct vs image space density, BUILD+SHAVE vs PIVOT

The object space element density statistics correspond to the expectation that BUILD+SHAVE should perform relatively worse than PIVOT as the possibility for intermediate feature generation breakdowns increases with increases in object density. The decrease in the gap between BUILD+SHAVE's performance and PIVOT's performance at high density may simply be due to the failure of both systems when buildings are in close proximity, and bad corner linkages are hard to avoid. The results of all four systems on the VILLAGE scenes (Figures B-325 through B-328) support this theory; none of the four

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**38**	7	-	-	-
BUILD+SHAVE	**20**	5	**22**	19	**23**	**20**
VHBUILD	5	3	3	3	9	6
PIVOT	19	3	12	**22**	12	18

Table 6-12: Relative performance summary, low image space element density

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**16**	5	-	-	-
BUILD+SHAVE	**13**	1	**10**	**11**	6	10
VHBUILD	3	3	1	2	3	1
PIVOT	8	5	8	**11**	**15**	**13**

Table 6-13: Relative performance summary, moderate image space element density

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0	**13**	5	-	-	-
BUILD+SHAVE	**8**	0	**6**	**6**	1	2
VHBUILD	2	1	0	3	4	3
PIVOT	5	1	4	**6**	**10**	**10**

Table 6-14: Relative performance summary, high image space element density

systems perform well on either of the two images, which have the highest object space element density of all 83 test areas.

We now turn to the image space element density statistics. Figures 6-41 through 6-44 give the relative building detection and quality percentage statistics in both 2D and 3D, as a function of the number of elements per 1000 pixels. In contrast to the object space density statistics, these figures show less clear trends with respect to image space element density, with a relatively even distribution of performance. PIVOT's performance is roughly worse in terms of the 2D statistics for high image space element densities, but it improves for the 3D performance metrics, and appears to be generally better in terms of 3D quality percentage for the high density images.

The performance summary statistics in Tables 6-12 through 6-14 show the same behavior, although the addition of the 3D branching factor statistics shows an interesting trend. In Table 6-12, which summarizes the results for the low density images, BUILD+SHAVE enjoys a clear advantage over PIVOT in terms of 3D branching factor. However, for moderate density images, Table 6-13 shows exactly the opposite trend, and for high density images, Table 6-14 shows that PIVOT has a superior branching factor in two-thirds of the images, with a corresponding improvement in 3D quality percentage.

It is harder to draw conclusions about image space element density, but the steady improvement in 3D branching factor for PIVOT as the image space element density increases suggests that PIVOT is better able to control the generation of false positive voxels as elements in the image become increasingly close. There are several factors which might be responsible, including PIVOT's use of overlap analysis to reject volumes which overlap other volumes with higher verification scores. As image space element density increases, the likelihood of generating multiple hypotheses covering overlapping areas of the image increases due to the likelihood of using nearby edges and corners to form multiple instances of building polygons. Only PIVOT incorporates overlap analysis to eliminate multiple coverage of portions of object space, and so it could be expected to exhibit an improved 3D branching factor in high density images.

The results of the complexity and density experiments can be summarized as follows:

- PIVOT exhibits better overall performance than BUILD+SHAVE for buildings of high complexity, where the use of primitives for object representation is better suited to represent complex structures.
- PIVOT exhibits better overall performance in scenes with increasing object space density than BUILD+SHAVE, consistent with the expectation that local grouping techniques for intermediate feature generation will begin to breakdown as feature density increases; BUILD+SHAVE exhibits better overall performance in scenes with lower object space density, consistent with BUILD+SHAVE's less stringent minimum data requirements for the generation of hypotheses (2 edges, against 3 for PIVOT).
- PIVOT shows steady improvement relative to other systems in 3D branching factor as the image space density increases, indicating the effectiveness of overlap analysis (Section 5.5.2) in reducing the number of false positive voxels.

In the image/scene complexity analyses thus far, the focus has been on identifying broad trends in system performance with respect to various properties of the image and scene. Such analyses are useful in coarsely categorizing system performance in various portions of the image/scene complexity space. However, it is also useful to assess performance on specific examples in detail, to understand the impact of various assumptions made by object detection and delineation algorithms. In particular, specific case studies highlight the points at which various algorithms fail to adhere to the six principles of Chapter 2. The next section examines issues in edge fragmentation, edge grouping, and object orientation with respect to vanishing point detection.

6.6. Detection and delineation performance case studies

The importance of maintaining all possible interpretations of image features until they can conclusively be discarded or accepted has been stressed throughout this work, most concisely by Principle 4: do not discard interpretations prematurely. It was also noted in the discussions of hypothesis generation and verification that this principle often proves difficult to follow, in the absence of constraints (such as vanishing points) which place no limitations on the scene, image, or scene model.

In this section, two case studies are presented which show the consequences of discarding interpretations prematurely. The first case study on edge fragmentation and edge grouping shows that the different edge processing approaches used by PIVOT and BUILD+SHAVE both fail to model particular classes of low level edge features, and can suffer performance problems as a result. The second case study considers the effect of vanishing point detection failures on PIVOT's performance in scenes with multiple dominant object orientations.

6.6.1. A case study: edge fragmentation and grouping

The discussion of Principle 4 in Chapter 2 gave examples of the deleterious effects of discarding interpretations prematurely. One of these examples revolved around the use of perceptual grouping methods which use local heuristics to determine whether image features are physically related. In this study, grouping methods are revisited, in the context of the different low-level edge processing methods that PIVOT and BUILD+SHAVE employ.

All four systems studied in this chapter use the Nevatia-Babu edge operator [78] to obtain raw edge data. However, PIVOT uses recursive line fitting on the raw edges to obtain straight line segments for vanishing point detection, while BUILD, BUILD+SHAVE, and VHBUILD all use a collinear line grouping procedure on the raw edge data to handle edge fragmentation due to poor image gradient or occluding features. The following experiments show that both approaches fail to strictly adhere to Principle 4, and that this failure translates to degraded object detection and delineation performance.

In the first experiment, PIVOT is run twice on image RADT11WOB, an image of a portion of Fort Hood, Texas. In one run, PIVOT uses its normal recursive line fitting procedure on the raw edge data. In the other run, PIVOT instead uses BUILD's collinear line grouping procedure on the raw edge data. All other aspects of PIVOT's algorithm are held constant across both runs.

Figure 6-45 shows the edge pixels of an edge detected along the roof perimeter of one wing of a rectilinear L-shaped building, after BUILD's collinear line grouping process has been run to reduce fragmentation effects. Here, the edge is preserved as a single entity. In Figure 6-46, the same edge pixels are shown, after PIVOT's recursive line fitting algorithm has been run. Note that the edge has now been broken into six segments, depicted by series of alternating white-on-black and black-on-white dots. This kind of edge fragmentation means that PIVOT will have difficulties extracting the full extent of the structure.

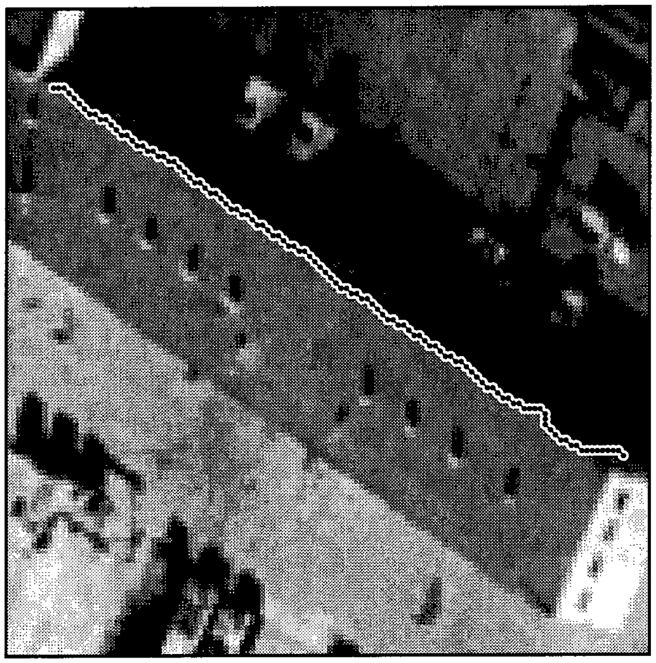


Figure 6-45: An edge after collinear line grouping, RADT11WOB

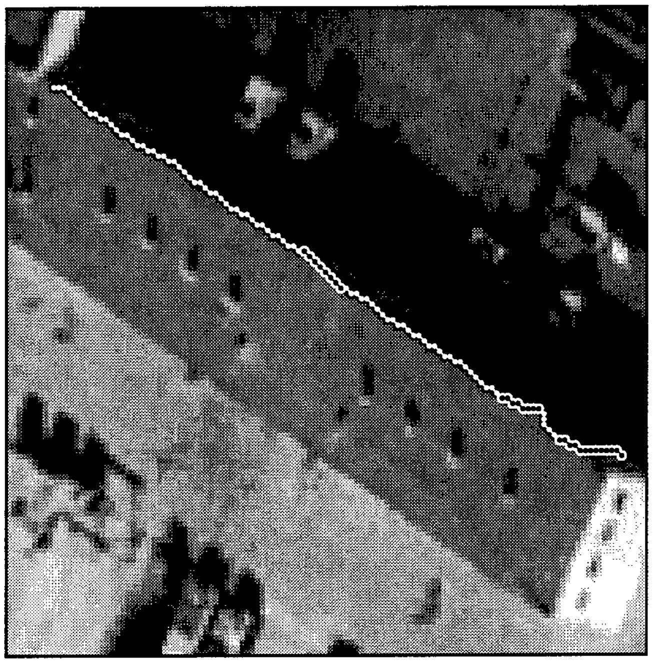


Figure 6-46: An edge after recursive line splitting, RADT11WOB

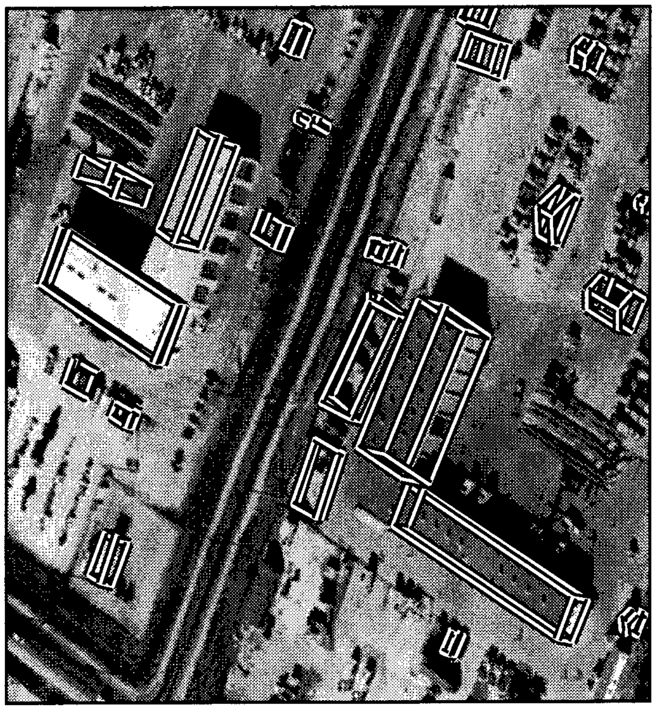


Figure 6-47: PIVOT results using collinear grouped edges, RADT11WOB

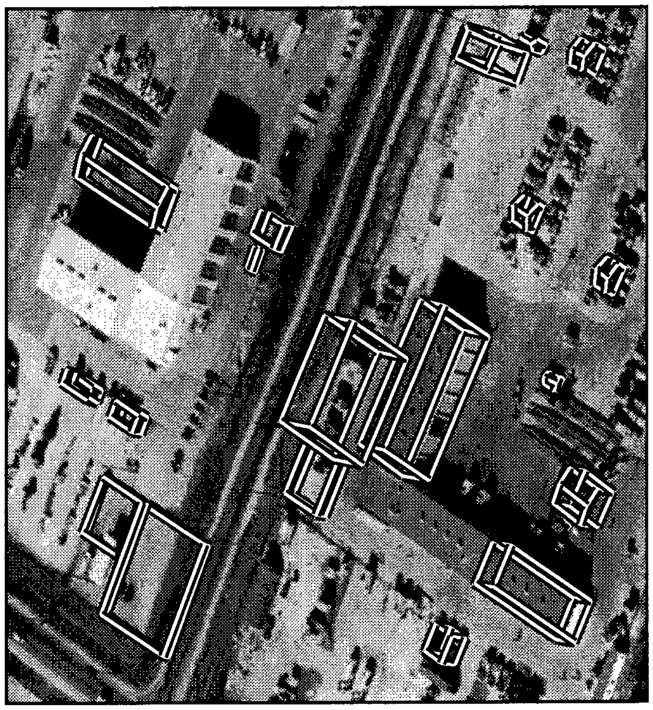


Figure 6-48: PIVOT results using recursively fit edges, RADT11WOB

Figure 6-47 shows the results of running PIVOT with BUILD's collinear grouped edges on the RADT11WOB area, and Figure 6-48 shows the results of running PIVOT with its standard set of recursively fit line segments. Using BUILD's edges, PIVOT is able to obtain at least a partial model for every wing of both L-shaped buildings, while it obtains only two partial models when using its standard

edges	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
recursive line fitting	34.67	2.899	17.29	29.60	2.627	16.65
collinear line grouping	71.34	0.979	42.01	52.54	0.989	34.58

Table 6-15: Comparison of PIVOT results using different edge methods, RADT11WOB

recursively fit line segments. Note also that the building shown in Figures 6-45 and 6-46 is in fact truncated in Figure 6-48. Table 6-15 gives the performance statistics on this scene for both PIVOT runs. PIVOT's numbers are significantly better for every performance metric on this image when using the collinear grouped edges in place of the recursively fit edges.

The results for RADT11WOB illustrate a situation where PIVOT has failed to generate the correct interpretation for several lines due to edge fragmentation, instead using partial line segments which cause shortening of many building models, and in some cases outright failure to generate a building hypothesis. However, simply adopting BUILD's processing approach for raw edge data can lead to problems as well, as the next example shows.

In the second experiment, PIVOT is again run twice, this time on a different image from Fort Hood, RADT5WOB. As before, one run uses the normal recursive line fitting procedure, while the other run uses BUILD's collinear line grouping algorithm, holding everything else constant.

Figure 6-49 shows the edge pixels of an edge detected along the two slanted ridges of a peaked roof building, after BUILD's collinear line grouping algorithm has been run. Here, as in the previous experiment, the edge is maintained as a single entity; but in this case, the edge actually merges two distinct edges in the scene. In Figure 6-50, the same edge pixels are shown, after PIVOT's recursive line fitting algorithm has been run. Here, recursive line fitting is able to break the edge into two segments, depicted by series of alternating white-on-black and black-on-white dots.

Figure 6-51 shows the results of running PIVOT with BUILD's collinear grouped edges on the RADT5WOB area, and Figure 6-52 shows the results of running PIVOT with the normal set of recursively fit line segments. Using BUILD's edges, PIVOT is unable to obtain vanishing points for the slanted ridges of the peaked roof buildings, and this is reflected in the approximations of the peaked roof buildings by flat roof buildings. Also, note that many of the flat roof building models are only partial, corresponding to "half" of a peaked roof building. In contrast, the regular PIVOT results in Figure 6-52 show that the peaked roof buildings are generally modeled correctly, since PIVOT was able to detect vanishing points for the slanted peak ridges. Table 6-16 shows the performance statistics for both cases, and here, PIVOT's standard edge processing method leads to significantly better detection and delineation performance over all six metrics.

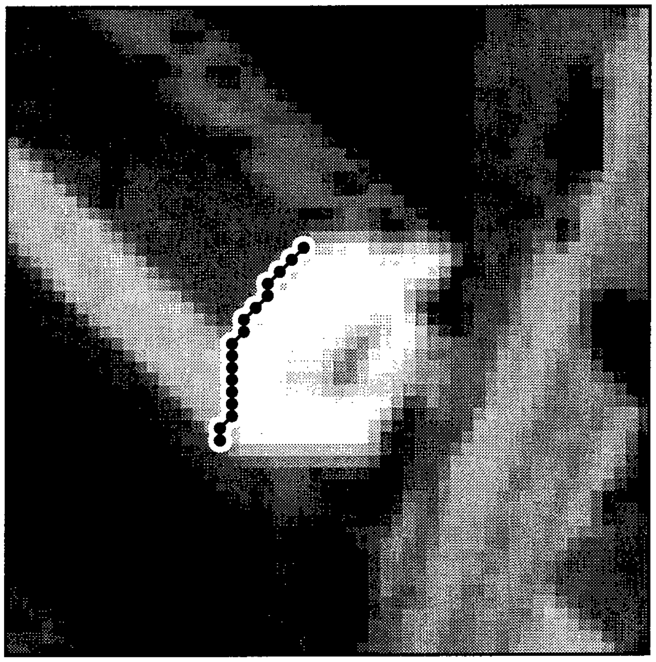


Figure 6-49: An edge after collinear line grouping, RADT5WOB

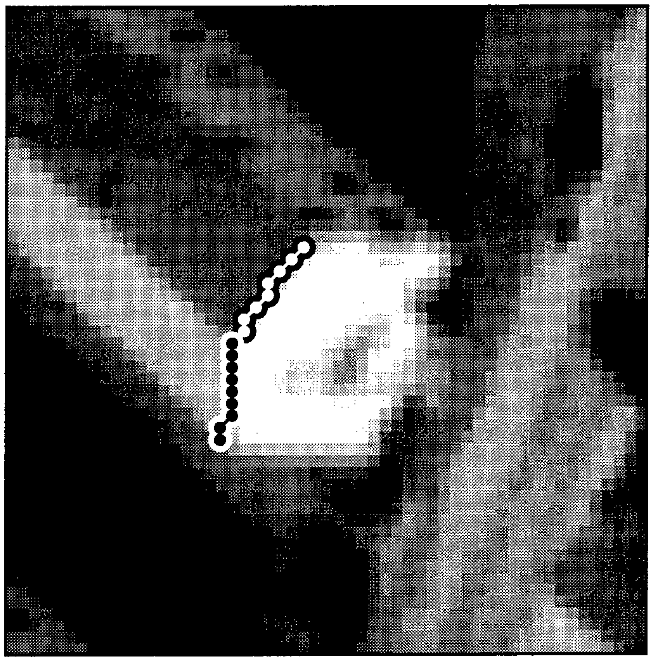


Figure 6-50: An edge after recursive line splitting, RADT5WOB

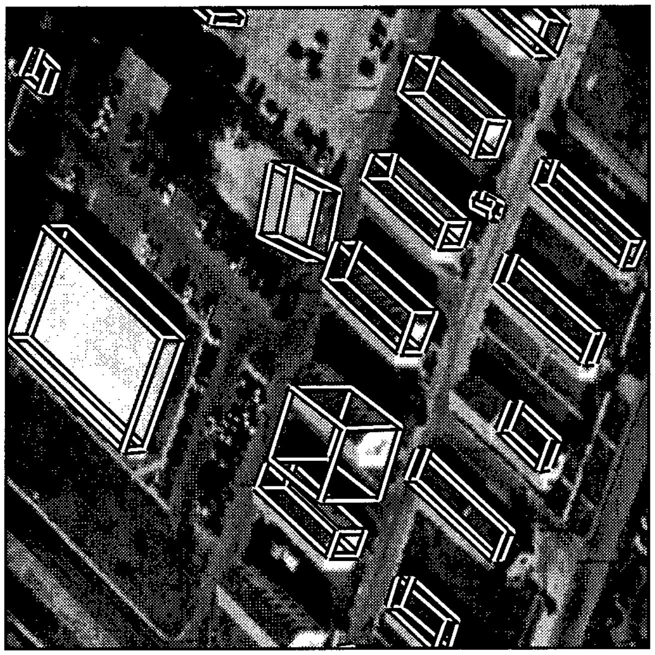


Figure 6-51: PIVOT results using collinear grouped edges, RADT5WOB

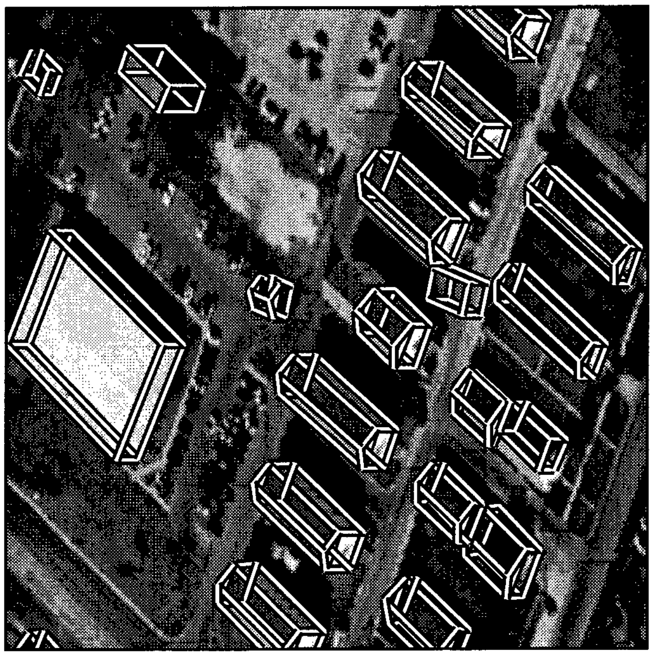


Figure 6-52: PIVOT results using recursively fit edges, RADT5WOB

This case study shows that each low-level edge processing approach has its benefits and drawbacks. In light of Principle 4, both methods are inadequate because they both discard interpretations of raw edge data prematurely. In BUILD's case, the grouping mechanism does not allow for the possibility that an edge may be comprised of multiple distinct physical edges in the world; in PIVOT's case, the recursive line fitting mechanism does not allow for the possibility that noise, and not actual changes in object boundaries, may be responsible for slight changes in the imaged edge orientation.

edges	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
recursive line fitting	84.25	0.256	69.28	76.97	0.292	62.84
collinear line grouping	60.10	0.691	42.47	50.74	0.911	34.70

Table 6-16: Comparison of PIVOT results using different edge methods, RADT5WOB

There are two possible approaches for handling this problem. One approach involves the maintenance of lineage information, as described in Section 5.2. PIVOT could be modified to use both edge processing methods, maintaining descendant pointers from the raw edge pixels to both the collinear grouped edges and the recursively fit edges, essentially placing these two interpretation of the raw edge data in competition. As with many of the intermediate feature generation phases, the difficulty with this approach lies in combinatorics; the number of edge interpretations to be considered would significantly increase.

The alternate approach is to handle fragmentation at the hypothesis extrusion phase. In its current implementation, PIVOT can extrude volumes vertically; it could be modified to extrude volumes in other directions as well. In the RADT11WOB examples, extrusion of the truncated rectangular wing along the horizontal vanishing lines would lead to a more complete building model. The difficulty with this approach is that it requires at least an initial building "seed" as a starting point; as Figure 6-48 showed, fragmentation can often be severe enough to prevent any building models from being generated at all.

This case study has illustrated one way in which premature removal of potentially legal interpretations of features significantly impacts performance. The next case study illustrates another way in which failure to follow Principle 4 can lead to significant degradation of performance.

6.6.2. A case study: object orientation

Chapter 3 presented a vanishing point detection method based on the use of primitive models to guide the search for vanishing points, and the use of edge error models to account for uncertainty in edge orientation. That method, incorporated in PIVOT, was shown to exhibit improved performance over classical vanishing point detection techniques. However, despite its ability to handle multiple pairs of orthogonal horizontals, this method can also break down when buildings lie in multiple orientations and have different shapes, in images with poor contrast.

The 83 test images used in this work typically have rectilinear structures which lie on a perpendicular grid. This makes vanishing point detection easier, as the orientations of object edges are shared, leading to more pronounced peaks in the Gaussian sphere histogram. In terms of the image/scene complexity space, the 83 images do not give a truly representative picture of *object orientation complexity*; we would certainly expect the vanishing point method to encounter difficulties where buildings lie in multiple orientations, such as in suburban areas around cul-de-sacs, near riverbanks, or in areas of significant terrain relief.

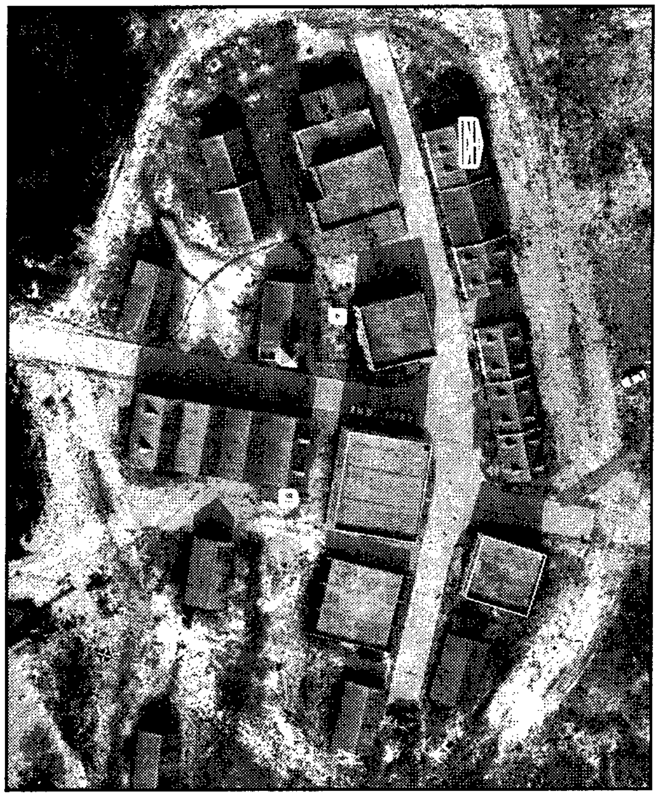


Figure 6-53: Results using automatically detected vanishing points, VILLAGE_FTBENN407

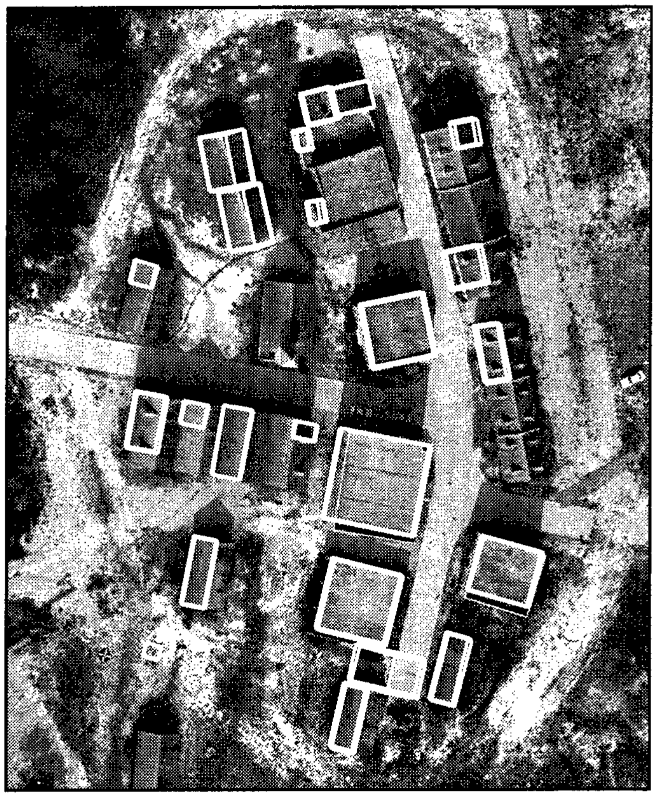


Figure 6-54: Results using manually measured vanishing points, VILLAGE_FTBENN407

vanishing points	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
automatic	1.06	0.545	1.05	0.71	1.124	0.70
manual	49.19	0.124	46.35	24.93	0.451	22.42

Table 6-17: Comparison of PIVOT results with manual vs auto vanishing points, VILLAGE_FTBENN407

The best examples of high object orientation complexity in the test imagery are the pair of VILLAGE images taken of a military training site in Fort Benning, Georgia. One of these images, VILLAGE_FTBENN407, is shown in Figure 6-53, along with the PIVOT results. These results are particularly poor; only three essentially accidental building models are generated, and out of the 83 test images, PIVOT produces its lowest 3D quality percentage on this scene (it should be noted that the other VILLAGE scene leads to PIVOT's fourth worst 3D quality percentage). The difficulties PIVOT encounters on this scene are comprised of several factors, including severe edge fragmentation, a lack of contrast across edge boundaries, a lack of good verticals, and shadows which are both difficult to distinguish from the ground and occluded by nearby buildings. However, one of the most significant factors is poor vanishing point detection performance. As in the FLAT_L example of Section 3.5, texture edges make significant contributions to the Gaussian sphere histogram, leading to spurious vanishing point detection.

To contrast the effects of poor vanishing point detection with robust vanishing point detection on PIVOT's performance, PIVOT was run again without the use of the automated vanishing point detection method, instead using a set of manually computed vanishing points. These vanishing points were derived from the edges in the ground truth models for this image by Equation (A.13) from Appendix A. All other aspects of PIVOT's processing phases were left unchanged.

Figure 6-54 shows the results of this second PIVOT run. The difference in performance is striking; even despite the edge fragmentation, poor contrast, and occluded shadows, PIVOT is able to capture a significant portion of the building objects in the scene, with a low number of false positives. Table 6-17 gives the performance statistics for both runs; not surprisingly, PIVOT's performance with manually derived vanishing points is much better than with automatically generated vanishing points.

There are several potential approaches for increasing the robustness of automated vanishing point detection; these are outlined elsewhere, in Section 3.7. This example illustrates the importance of obtaining good vanishing point solutions; failure to generate accurate vanishing points can be regarded as discarding the correct orientation interpretations for the edges of objects. It is clear, after this study and the earlier study on edge fragmentation and grouping, that Principle 4 captures a crucial aspect of object detection and delineation which cannot be neglected if robust performance is to be achieved.

6.7. Conclusions

In this chapter, a comprehensive comparative analysis of PIVOT and three other building extraction systems was presented. This analysis, carried out on an extensive dataset of 83 images, was supplemented by broad experiments to assess the impact of image obliquity, object complexity, and object density on detection and delineation performance. Case studies illustrated the impact of low-level edge processing methods and vanishing point detection breakdowns on PIVOT's performance.

Several conclusions can be drawn from this analysis:

- PIVOT's comparative improvements in 3D object detection and delineation, as supported by its superior performance on a majority of the test scenes in terms of 3D building detection rate, branching factor, and quality percentage, validates the central claim of this thesis: *basic volumetric forms for object recognition can be detected, delineated, and verified with cues derived from rigorous modeling of the image acquisition process.*
- PIVOT shows improved performance on images of increasingly high effective obliquity, demonstrating the utility of a rigorous camera model in conjunction with 3D modeling of object and shadow geometry to operate on images with a wide range of viewing angles. These improvements are the direct result of following Principles 1, 3, and 5.
- PIVOT exhibits improvements in 3D object detection and delineation on structures of high complexity and density, consistent with its augmented object modeling capabilities. The use of primitives gives PIVOT increased representational power, in keeping with Principle 2.
- The comparative performance results in this chapter show that different assumptions about the scene, image, and scene model lead to different performance in different parts of the image/scene complexity space. These results further support the "cooperative methods"

paradigm for object detection and delineation: no one technique is likely to achieve success across the entire complexity space [93]. PIVOT's improved 3D performance supports the cooperative methods paradigm, as it uses multiple complementary techniques in 3D hypothesis generation (vertical and shadow based height estimation) and verification (shadow consistency, edge gradient, intensity homogeneity, and object illumination models).

- The case studies on edge fragmentation, grouping, and object orientation complexity highlight areas of the complexity space where cooperative methods and new techniques are needed to achieve robust performance. Images such as VILLAGE_FTBENN407 present formidable difficulties for current state-of-the-art monocular building extraction systems.

PIVOT extends the state-of-the-art in object detection and delineation along a number of different axes; the final chapter summarizes these improvements, and briefly discusses several avenues for further work based on the results of this thesis.

Chapter 7
Conclusions

This chapter briefly surveys the research accomplishments of the thesis as a whole, and addresses various issues raised during the course of the work in a section on future research.

7.1. Research accomplishments

This thesis has presented and developed several original ideas for 3D object detection and delineation, and implemented these ideas in the task domain of building extraction from monocular aerial imagery. The main hypothesis of this work, that rigorous modeling of the image geometry leads to improved object detection and delineation performance, was explored through the design and implementation of a fully automated building extraction system. To assess the performance of this system in 2D image space and 3D object space, a comparative performance evaluation methodology was developed and applied to a large and varied set of test imagery. Each of these contributions are outlined in the following sections.

7.1.1. Photogrammetric modeling for improved performance

The hypothesis that rigorous photogrammetric modeling leads to improved detection and delineation of basic volumetric forms for object recognition served as the basis for six object detection and delineation principles, developed in Chapter 2. The first principle argued for the use of photogrammetric methods to model the image acquisition process; it is this approach that allows many of the remaining principles to be followed, resulting in superior 3D detection and delineation performance on complex scenes.

In accordance with Principle 2, the use of simple primitive volumetric models for a generic representation of manmade structures in aerial images alleviates the need for a large model library and gains the ability to flexibly model complex shapes by composition. Instances of these generic models can be created through the use of new techniques for automatic detection of vanishing points in complex imagery, using knowledge of object shape to guide the vanishing point search, and new models of edge orientation uncertainty to provide robust performance in the presence of significant noise.

In keeping with Principles 3 and 4, the use of methods for exploiting vanishing point information as constraints on the construction of geometrically legal primitives from intermediate features significantly reduces the combinatorics of hypothesis search without discarding potentially legal primitives. In accordance with Principle 5, the use of projective geometry in conjunction with a rigorous photogrammetric camera model provides the ability to analyze shadow shape and object photometry for hypothesis generation (shadow-based extrusion) and verification (shadow-based and illumination-based consistency testing of object models).

7.1.2. Implementation of an automated building extraction system

To explore the central hypothesis of this work, a fully automated monocular building extraction system was implemented, based on photogrammetric camera modeling, which generates 3D object space building models for complex scenes from a wide range of viewpoints. PIVOT (**P**erspective **I**nterpretation of **V**anishing points for **O**bjects in **T**hree dimensions) integrates primitive object representations, vanishing point constraints, object space modeling of primitive and shadow geometry, and multiple complementary verification methods to generate 3D object space building models for a wide variety of aerial scenes.

PIVOT accepts panchromatic mapping photography and the image acquisition parameters as input, and produces geographically referenced 3D building wireframes, without manual intervention or parameter tuning. PIVOT uses a variety of techniques in the processing flow from corners to 2-corners to primitives to 3D object space building models, many of which make heavy use of projective geometry and vanishing point information, derived from the camera model, to construct intermediate features. This approach stands in contrast to the techniques used by many existing systems which make limited or no use of knowledge about the image acquisition process.

7.1.3. Development and application of performance evaluation methodology

To test the central hypothesis of this work, a quantitative comparative evaluation methodology for object detection and delineation was developed, using unbiased performance metrics on large imagery datasets, with specific analyses to address the impact of image and scene complexity. Using this methodology, PIVOT was compared to three existing building extraction systems on 83 test images covering a wide variety of geographical areas, object complexities, and viewing angles.

This comparative performance evaluation, the most extensive of its kind to date, was supplemented by analyses of the effects of object complexity, density, and image obliquity, illustrating PIVOT's improved modeling performance from highly oblique viewpoints and on complex structures, as well as case studies to highlight the impact of low level edge processing and object orientation on particular detection and delineation algorithms.

The research contributions of this thesis serve as an illustration of the benefits of photogrammetric modeling for a computer vision task in a difficult domain. In addition to providing the ability to model objects in world coordinates, photogrammetry also facilitates the use of intrinsic constraints derived from the imaging process for object detection and delineation.

7.2. Future research

The results and contributions of this thesis suggest several new avenues for future research efforts. This section gives brief sketches of a few interesting directions for future work, beginning with topics specific to the monocular object detection and delineation problem, and expanding to increasingly broad topics.

7.2.1. Constraints, heuristics, and combinatorics

The interplay between constraints, heuristics, and combinatorics in intermediate feature generation merits further study. Many of the heuristics PIVOT used in hypothesis evaluation and verification were useful for capturing qualitative aspects of object photometry and geometry, but the performance of these heuristics falls short when compared to intrinsic constraints, such as the vanishing point constraints derived from the image acquisition parameters and primitive shape. Intrinsic constraints, such as the vanishing point geometric constraints of Chapter 4, have the desirable property that only illegal feature combinations are discarded when they are employed, unlike heuristics which have no such guarantee. However, heuristics are frequently necessary in the absence of intrinsic constraints, to mediate the combinatorial problems inherent in hypothesis search and construction. Finding new intrinsic constraints, or, more likely, tighter heuristics for constraining the hypothesis search space remains an important research topic, particularly in the analysis of object photometry.

This interplay of constraints, heuristics, and combinatorics is also intimately tied to the depth and breadth of the feature hierarchy. As mentioned earlier, 2-corners are only one possible intermediate feature combining three edges; another is the trihedral vertex. In the spirit of the *cooperative methods* approach to feature extraction, it would be reasonable to maintain both intermediate features in the feature hierarchy, but this addition also increases the number of legal interpretations of low-level image features. While Principle 4 dictates that both features be generated to avoid discarding interpretations prematurely, mediating the combinatorial consequences of adding new features is equally important. An analysis of the combinatorial effects of expanding the intermediate feature hierarchy would be an important first step in increasing the robustness of object detection and delineation algorithms.

7.2.2. The role of domain knowledge

It has been well understood that domain-specific knowledge is often an integral component of good performance in vision tasks. In PIVOT's case, the aerial image analysis domain provided two important constraints; first, the assumption that objects sit on a "ground" plane, removing one degree of freedom from the vanishing point search and increasing its efficiency, and second, the assumption that the majority of objects can modeled by symmetric primitives, due to the pervasiveness of symmetry in manmade constructions [105]. Understanding how the object detection and delineation methods presented in this thesis translate to other task domains is an open question.

On a related note, the addition of new primitives and the development of more powerful methods for attachment and extension also merits further study. PIVOT currently uses two primitives, with simple attachment and extension schemes. Section 3.7 discussed issues in vanishing point detection for increasingly complex primitive shapes, and Section 5.6 briefly addressed the possibilities for more flexible attachment schemes and multi-directional primitive extension methods. As in feature generation, the principal difficulty in these cases is combinatorial in nature; finding vanishing point patterns for complex shapes on the Gaussian sphere, allowing arbitrary attachments of primitive volumes, or extending primitives along multiple vanishing lines requires a significant increase in computation time. New developments in any of these areas would significantly enhance the modeling power of a system such as PIVOT.

7.2.3. Multi-image analysis algorithms

PIVOT is a monocular system by design, to assess the performance of object detection and delineation methods in the limiting case of a single image. It should be clear that even assuming best-case performance by PIVOT, some image effects will still require binocular stereo or multi-image analysis to disambiguate uncertainty in 3D position and orientation of scene features. Recent work on multi-image matching [89, 19, 81] and adaptive vergence for object space stereo over widely varying viewpoints [18] has illustrated the potential of stereo approaches for feature extraction. One of the current pitfalls of many multi-image matchers are the combinatorics of finding good feature matches as the number of images increases. It is natural to ask whether the intrinsic constraints used by PIVOT might also apply to the general problem of multi-image matching, alleviating the combinatorial problems associated with feature matching while gaining the increased solution accuracy and precision of multi-image solutions.

Conversely, multi-image matching has a role to play in correctly handling occlusions of objects and shadows. Currently, PIVOT makes no provision for building-building occlusion, building-shadow occlusion, or shadow-building occlusion. A key difficulty in correctly deciding when occlusions have occurred is knowing whether the occluded or occluding object actually exists, and is not the result of a geometrically and photometrically plausible, but accidental, alignment of image features. Single images do not always provide enough information to provide conclusive verification, but even rough disparity or elevation maps produced via multi-image analysis might suffice to determine whether an occluding or occluded object is plausible, and allow hypothesis generation and verification to proceed in these cases. Mechanisms for reasoning about occlusion will be essential for the automated construction of models from highly oblique views where object-object occlusion is common, and for complex objects with overlapping and self-occluding parts.

7.2.4. Emerging applications

The construction of cartographic databases for distributed simulation is currently an active topic of research, and the ability to rapidly and accurately model manmade features in geographic sites of interest is an important aspect of simulation research efforts [60]. An important question revolves around the tradeoff between simulation accuracy and modeling effort in database construction. Consider two simulation databases constructed for a small portion of Fort Hood, Texas, depicted in Figures 7-1 and 7-2. The database in Figure 7-1 contains models generated semi-automatically from SITECITY; the database in Figure 7-2 contains models generated fully automatically by PIVOT. All other aspects of the databases are identical; the road network generation, terrain skin generation, and feature integration techniques have been described in detail elsewhere [70, 86, 109].

While the automatically generated building models in Figure 7-2 have several false positive buildings, the primary qualitative features (rows of barracks and an L-shaped building) are preserved, without any manual editing. An interesting question from the simulation standpoint is whether the manual effort required to generate cartographically accurate models, like those shown in Figure 7-1, is worth the increase in simulation fidelity; the answer to this question lies in the performance demands placed on a particular simulation task. From a research standpoint, another interesting question is whether fully automated hypothesis generation, followed by the use of semi-automated site modeling tools such as those offered by SITECITY, provides faster construction of site models than semi-automated systems alone while maintaining the accuracy provided by such tools. A third question is whether the computer vision algorithms employed by PIVOT could also be employed by semi-automated systems such as SITECITY, to increase usability and site modeling efficiency.

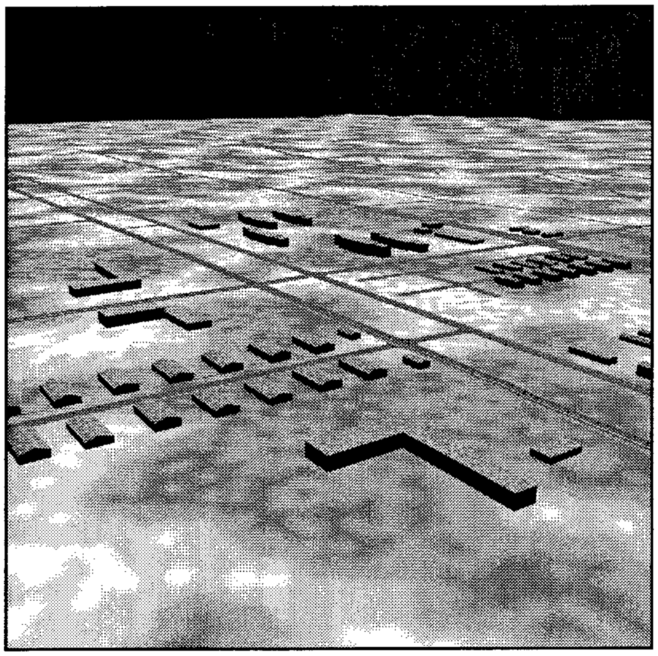


Figure 7-1: Simulation database with semi-automatically constructed buildings

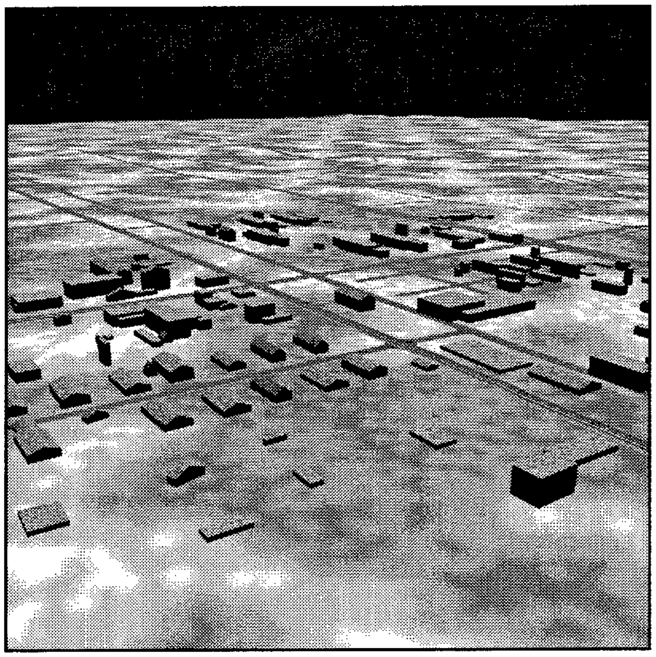


Figure 7-2: Simulation database with automatically constructed buildings

As has been noted several times in this work, the use of photogrammetric methods not only provides constraints for object detection and delineation, but also provides the ability to operate directly in object space. The importance of this ability cannot be understated, as it permits the use of disparate data sources in a consistent setting. While this thesis focused on panchromatic frame mapping photography, other sensors such as multispectral and hyperspectral scanners provide complementary kinds of data, measuring scene properties outside the visible spectrum. Merging the results of surface material classification and multi-image analysis with object detection and delineation algorithms provides a combined data source unavailable from any single imaging device [26, 27]. Developing improved methods for exploiting multiple disparate sources of data to refine 3D site models is another important research topic for the future.

On a final related note, rigorous camera modeling enabled PIVOT to establish new levels of performance in new portions of the image/scene complexity space. As the demands for timely compilation of urban and suburban geographic models increase, photogrammetric methods and the constraints they provide can be expected to play a key role in the development of object detection and delineation methods to meet these demands.

Bibliography

[1] Agin, G. J. and Binford, T. O.
Computer Description of Curved Objects.
IEEE Transactions on Computers C-25(4):439-449, 1976.

[2] Barakat, H., Doucette, P., and Mikhail, E. M.
Photogrammetric Analysis of Image Invariance.
In Ebner, H., Heipke, C., and Eder, K. (editors), *Proceedings: ISPRS Commission III Symposium on Spatial Information from Digital Photogrammetry and Computer Vision, Volume 30, Part 3/1*, pages 25-34. Munich, Germany, 1994.

[3] Barakat, H., Weerawong, K., and Mikhail, E. M.
Comparison Between Invariance and Photogrammetry for Image and Object Transfer.
In McKeown, D. M and Dowman, I. J. (editors), *Proceedings: SPIE Conference on Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, Volume 2486*, pages 13-24. April 19-21, 1995.

[4] Barnard, S.
Interpreting Perspective Images.
Artificial Intelligence 21:435-462, 1983.

[5] Bergevin, R. and Levine, M. D.
Generic Object Recognition: Building and Matching Coarse Descriptions from Line Drawings.
IEEE Transactions on Pattern Analysis and Machine Intelligence 15(1):19-36, January, 1993.

[6] Besl, P. J. and Jain, R. C.
Three-Dimensional Object Recognition.
Computing Surveys 17(1):75-145, March, 1985.

[7] Biederman, I.
Human Image Understanding: Recent Research and a Theory.
Computer Vision, Graphics, and Image Processing 32:29-73, 1985.

[8] Binford, T. O.
Visual Perception by Computer.
In *Proceedings: IEEE Conference on Systems and Control.* December, 1971.

[9] Binford, T. O.
Survey of Model-Based Image Analysis Systems.
International Journal of Robotics Research 1(1):18-64, 1982.

[10] Borgefors, G.
Distance Transformations in Digital Images.
Computer Vision, Graphics, and Image Processing 34:344-371, 1986.

[11] Braun, C.
Interpretation and Correction of Single Line Drawings for the Reconstruction of Objects in Space.
In Ebner, H., Heipke, C., and Eder, K. (editors), *Proceedings: ISPRS Commission III Symposium on Spatial Information from Digital Photogrammetry and Computer Vision, Volume 30, Part 3/1*, pages 85-90. Munich, Germany, 1994.

[12] Brillault-O'Mahony, B.
New Method for Vanishing Point Detection.
Computer Vision, Graphics, and Image Processing: Image Understanding 54(2):289-300, 1991.

[13] Bro-Nielsen, M.
Building Detection in Aerial Images.
Master's thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, August, 1992.

[14] Brooks, R. A.
Model-Based 3-D Interpretation of 2-D Images.
IEEE Transactions on Pattern Analysis and Machine Intelligence 5(2):140-150, March, 1983.

[15] Brown, C. M.
Inherent Bias and Noise in the Hough Transform.
IEEE Transactions on Pattern Analysis and Machine Intelligence 5(5):493-505, September, 1983.

[16] Chin, R. T. and Dyer, C. R.
Model-Based Recognition in Robot Vision.
Computing Surveys 18(1):67-108, March, 1986.

[17] Clowes, M. B.
On Seeing Things.
Artificial Intelligence 2(1):79-116, 1971.

[18] Cochran, S. D.
Adaptive Vergence for the Stereo Matching of Oblique Imagery.
In *Proceedings: ARPA Image Understanding Workshop, Volume II*, pages 1335-1348. Monterey, California, November, 1994.

[19] Collins, R. T.
A Space-Sweep Approach to True Multi-Image Matching.
In *Proceedings: ARPA Image Understanding Workshop, Volume II*, pages 1213-1220. Palm Springs, California, February, 1996.

[20] Collins, R. T. and Weiss, R. S.
Vanishing Point Calculation as a Statistical Inference on the Unit Sphere.
In *Proceedings: Third International Conference on Computer Vision*, pages 400-403. 1990.

[21] Collins, R. T., Jaynes, C., Stolle, F., Wang, X., Cheng, Y.-Q., Hanson, A. R., and Riseman, E. M.
A System for Automated Site Model Acquisition.
In McKeown, D. M and Dowman, I. J. (editors), *Proceedings: SPIE Conference on Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, Volume 2486*, pages 244-254. April 19-21, 1995.

[22] Collins, R. T., Hanson, A. R., Riseman, E. M., Jaynes, C., Stolle, F., Wang, X., and Cheng, Y.-Q.
UMass Progress in 3D Building Model Acquisition.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 305-315. Palm Springs, California, February, 1996.

[23] Conners, R. W., Trivedi, M. M., and Harlow, C. A.
Segmentation of a High-Resolution Urban Scene Using Texture Operators.
Computer Vision, Graphics, and Image Processing 25:273-310, 1984.

[24] Dickinson, S. J., Pentland, A. P., and Rosenfeld, A.
From Volumes to Views: An Approach to 3-D Object Recognition.
Computer Vision, Graphics, and Image Processing: Image Understanding 55(2):130-154, March, 1992.

[25] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F.
Computer Graphics: Principles and Practice, 2nd ed.
Addison-Wesley, Reading, MA, 1990.

[26] Ford, S. J. and McKeown, D. M.
Information Fusion of Multispectral Imagery for Cartographic Feature Extraction.
In *Proceedings: DARPA Image Understanding Workshop*, pages 805-820. January, 1992.

[27] Ford, S. J. and McKeown, D. M.
Performance Evaluation of Multispectral Analysis for Surface Material Classification.
In *Proceedings of the International Geoscience and Remote Sensing Symposium*, pages 2112-2116. August, 1994.

[28] Fritsch, D., Sester, M., and Schenk, T.
Test on Image Understanding.
In Ebner, H., Heipke, C., and Eder, K. (editors), *Proceedings: ISPRS Commission III Symposium on Spatial Information from Digital Photogrammetry and Computer Vision, Volume 30, Part 3/1*, pages 243-248. Munich, Germany, 1994.

[29] Fua, P.
Cartographic Applications of Model-Based Optimization.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 409-419. Palm Springs, California, February, 1996.

[30] Fua, P. and Hanson, A. J.
An Optimization Framework for Feature Extraction.
Machine Vision and Applications 4(2):59-87, 1991.

[31] Grimson, W. E. L., with Lozano-Perez, T. and Huttenlocher, D. P.
Object Recognition by Computer: The Role of Geometric Constraints.
MIT Press, Cambridge, MA, 1990.

[32] Grimson, W. E. L. and Lozano-Perez, T.
Model-Based Recognition and Localization from Sparse Range or Tactile Data.
International Journal of Robotics Research 3(3):3-35, 1984.

[33] Hanson, A. J., Pentland, A. P., and Quam, L. H.
Design of a Prototype Interactive Cartographic Display and Analysis Environment.
In *Proceedings: DARPA Image Understanding Workshop*, pages 475-482. February, 1987.

[34] Hartley, R. I.
Euclidean Reconstruction from Uncalibrated Views.
In Mundy, J. L., Zisserman, A., and Forsyth, D. (editors), *Applications of Invariance in Computer Vision: Second Joint European-US Workshop*, pages 237-256. Springer-Verlag, Porta Delgada, Azores, Portugal, 1993.

[35] Harwood, D., Chang, S., and Davis, L.
Interpreting Aerial Photographs by Segmentation and Search.
In *Proceedings: DARPA Image Understanding Workshop*, pages 507-520. February, 1987.

[36] Herman, M. and Kanade, T.
Incremental Reconstruction of 3D Scenes from Multiple, Complex Images.
Artificial Intelligence 30:289-341, 1986.

[37] Heuel, S. and Nevatia, R.
Including Interaction in an Automated Modeling System.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 429-434. Palm Springs, California, February, 1996.

[38] Horaud, R.
New Methods for Matching 3-D Objects with Single Perspective Views.
IEEE Transactions on Pattern Analysis and Machine Intelligence 9(3):401-412, May, 1987.

[39] Horn, B. K. P.
Extended Gaussian Images.
Proceedings of the IEEE 72(12):1671-1686, December, 1984.

[40] Hsieh, Y.
Design and Evaluation of a Semi-Automated Site Modeling System.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 435-459. Palm Springs, California, February, 1996.

[41] Hsieh, Y.
SiteCity: A Semi-Automated Site Modeling System.
In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 499-506. June, 1996.

[42] Huertas, A. and Nevatia, R.
Detecting Buildings in Aerial Images.
Computer Vision, Graphics, and Image Processing 41:131-152, April, 1988.

[43] Huffman, D. A.
Impossible Objects as Nonsense Sentences.
In Meltzer, B. and Michie, D. (editors), *Machine Intelligence 6*, pages 295-323. Edinburgh University Press, Edinburgh, 1971.

[44] Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J.
Comparing Images Using the Hausdorff Distance.
IEEE Transactions on Pattern Analysis and Machine Intelligence 15(9):850-863, September, 1993.

[45] Ikeuchi, K.
Generating an Interpretation Tree from a CAD Model for 3D-Object Recognition in Bin-Picking Tasks.
International Journal of Computer Vision 1(2):145-165, 1987.

[46] Irvin, R. B. and McKeown, D. M.
Methods for Exploiting the Relationship Between Buildings and Their Shadows in Aerial Imagery.
IEEE Transactions on Systems, Man, and Cybernetics 19(6):1564-1575, November, 1989.

[47] Jaynes, C., Stolle, F., and Collins, R.
Task Driven Perceptual Organization for Extraction of Rooftop Polygons.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 359-365. November, 1994.

[48] Kanade, T.
Recovery of the Three-Dimensional Shape of an Object from a Single View.
Artificial Intelligence 17:409-460, 1981.

[49] Kanatani, K.
Hypothesizing and Testing Geometric Properties of Image Data.
Computer Vision, Graphics, and Image Processing: Image Understanding 54(3):349-357, November, 1991.

[50] Kanatani, K.
Statistical Analysis of Focal-Length Calibration Using Vanishing Points.
IEEE Transactions on Robotics and Automation 8(6):767-775, December, 1992.

[51] Kanatani, K.
Statistical Analysis of Geometric Computation.
Computer Vision, Graphics, and Image Processing: Image Understanding 59(3):286-306, May, 1994.

[52] Kang, S. B. and Ikeuchi, K.
The Complex EGI: A New Representation for 3-D Pose Determination.
IEEE Transactions on Pattern Analysis and Machine Intelligence 15(7):707-721, July, 1993.

[53] Karara, H. M. (editor).
Non-Topographic Photogrammetry.
American Society for Photogrammetry and Remote Sensing, 1989.

[54] Kass, M., Witkin, A., and Terzopoulos, D.
Snakes: Active Contour Models.
International Journal of Computer Vision 1(4):321-331, 1987.

[55] Lin, C. and Nevatia, R.
Buildings Detection and Description from Monocular Aerial Images.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 461-468. Palm Springs, California, February, 1996.

[56] Lin, C., Huertas, A., and Nevatia, R.
Detection of Buildings Using Perceptual Grouping and Shadows.
In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 62-69. June, 1994.

[57] Liow, Y.-T. and Pavlidis, T.
Use of Shadows for Extracting Buildings in Aerial Images.
Computer Vision, Graphics, and Image Processing 49:242-277, 1990.

[58] Lowe, D. G.
Perceptual Organization and Visual Recognition.
Kluwer, Boston, 1985.

[59] Lueker, G. S.
A Data Structure for Orthogonal Range Queries.
In *Proceedings: 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 28-34. 1978.

[60] Lukes, G.
Cartographic Support for Advanced Distributed Simulation.
In McKeown, D. M and Dowman, I. J. (editors), *Proceedings: SPIE Conference on Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, Volume 2486*, pages 176-185. April 19-21, 1995.

[61] Lutton, E., Maitre, H., and Lopez-Krahe, J.
Contribution to the Determination of Vanishing Points Using Hough Transform.
IEEE Transactions on Pattern Analysis and Machine Intelligence 16(4):430-438, April, 1994.

[62] Mackworth, A. K.
Interpreting Pictures of Polyhedral Scenes.
Artificial Intelligence 4(2):121-137, June, 1973.

[63] Magee, M. J. and Aggarwal, J. K.
Determining Vanishing Points from Perspective Images.
Computer Vision, Graphics, and Image Processing 26:256-267, 1984.

[64] Maitre, H.
Contribution to the Prediction of Performances of the Hough Transform.
IEEE Transactions on Pattern Analysis and Machine Intelligence 8(5):669-674, September, 1986.

[65] McGlone, J. C.
Design and Implementation of an Object-Oriented Photogrammetric Toolkit.
In *International Archives of Photogrammetry and Remote Sensing, Volume XXIX, B2*, pages 334-338. 1992.

[66] McGlone, J. C.
Bundle Adjustment with Object Space Geometric Constraints for Site Modeling.
In McKeown, D. M. and Dowman, I. J. (editors), *Proceedings: SPIE Conference on Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, Volume 2486*, pages 25-36. April 19-21, 1995.

[67] McGlone, J. C. and Shufelt, J. A.
Projective and Object Space Geometry for Monocular Building Extraction.
In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 54-61. June, 1994.

[68] McKeown, D. M.
Toward Automatic Cartographic Feature Extraction.
In L. F. Pau (editor), *NATO ASI Series.* Volume F 65: *Mapping and Spatial Modelling for Navigation*, pages 149-180. Springer-Verlag, Berlin Heidelberg, 1990.

[69] McKeown, D. M. and Denlinger, J. L.
Map-Guided Feature Extraction from Aerial Imagery.
In *Proceedings: IEEE Workshop on Computer Vision: Representation and Control*, pages 205-213. Annapolis, MD, 1984.

[70] McKeown, D. M. and Denlinger, J. L.
Cooperative Methods for Road Tracking in Aerial Imagery.
In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 662-672. June, 1988.

[71] McKeown, D. M. and McGlone, J. C.
Integration of Photogrammetric Cues into Cartographic Feature Extraction.
In Barrett, E. B. and McKeown, D. M. (editors), *Proceedings: SPIE Conference on Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision, Volume 1944*, pages 2-15. April 14-15, 1993.

[72] McKeown, D. M., Harvey, W. A., and McDermott, J.
Rule-Based Interpretation of Aerial Imagery.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7(5):570-585, September, 1985.

[73] McLean, G. F. and Kotturi, D.
Vanishing Point Detection by Line Clustering.
IEEE Transactions on Pattern Analysis and Machine Intelligence 17(11):1090-1095, November, 1995.

[74] Mohan, R. and Nevatia, R.
Using Perceptual Organization to Extract 3-D Structures.
IEEE Transactions on Pattern Analysis and Machine Intelligence 11(11):1121-1139, November, 1989.

[75] Mundy, J. L. and Zisserman, A. (editors).
Geometric Invariance in Computer Vision.
MIT Press, Reykjavik, Iceland, March 25-28.

[76] Mundy, J. L., Zisserman, A., and Forsyth, D. (editors).
Applications of Invariance in Computer Vision: Second Joint European-US Workshop.
Springer-Verlag, Porta Delgada, Azores, Portugal, October 9-14.

[77] Nagao, M. and Matsuyama, T.
A Structural Analysis of Complex Aerial Photographs.
Plenum, New York, 1980.

[78] Nevatia, R. and Babu, K. R.
Linear Feature Extraction and Description.
Computer Graphics and Image Processing 13:257-269, July, 1980.

[79] Nevatia, R. and Binford, T. O.
Description and Recognition of Curved Objects.
Artificial Intelligence 8:77-98, 1977.

[80] Nicolin, B. and Gabler, R.
A Knowledge-Based System for the Analysis of Aerial Images.
IEEE Transactions on Geoscience and Remote Sensing GE-25(3):317-329, May, 1987.

[81] Noronha, S. and Nevatia, R.
Detection and Description of Buildings from Multiple Aerial Images.
In *Proceedings: ARPA Image Understanding Workshop, Volume I*, pages 469-478. Palm Springs, California, February, 1996.

[82] Parodi, P. and Piccioli, G.
3D Shape Reconstruction by Using Vanishing Points.
IEEE Transactions on Pattern Analysis and Machine Intelligence 18(2):211-217, February, 1996.

[83] Pentland, A. P.
Perceptual Organization and the Representation of Natural Form.
Artificial Intelligence 28:293-331, 1986.

[84] Pentland, A. P.
Automatic Extraction of Deformable Part Models.
International Journal of Computer Vision 4:107-126, 1990.

[85] Plantinga, H. and Dyer, C. R.
Visibility, Occlusion, and the Aspect Graph.
International Journal of Computer Vision 5(2):137-160, 1990.

[86] Polis, M. F., Gifford, S. J., and McKeown, D. M.
Automating the Construction of Large-Scale Virtual Worlds.
IEEE Computer 28(7):57-65, July, 1995.

[87] Preparata, F. P. and Shamos, M. I.
Computational Geometry: An Introduction.
Springer-Verlag, New York, 1990.

[88] Quan, L. and Mohr, R.
Determining Perspective Structures Using Hierarchical Hough Transform.
Pattern Recognition Letters 9(4):279-286, 1989.

[89] Roux, M. and McKeown, D. M.
Feature Matching for Building Extraction from Multiple Views.
In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 46-53. June, 1994.

[90] Shafer, S. A.
Shadows and Silhouettes in Computer Vision.
Kluwer, Boston, 1985.

[91] Shufelt, J. A.
Performance Evaluation and Analysis of Vanishing Point Detection Techniques.
In *Proceedings: ARPA Image Understanding Workshop, Volume II*, pages 1113-1132. Palm Springs, California, February, 1996.

[92] Shufelt, J. A.
Exploiting Photogrammetric Methods for Building Extraction in Aerial Images.
In *International Archives of Photogrammetry and Remote Sensing, Volume XXXI, B6/S*, pages 74-79. July, 1996.

[93] Shufelt, J. A. and McKeown, D. M.
Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery.
Computer Vision, Graphics and Image Processing: Image Understanding 57(3):307-330, May, 1993.

[94] Slama, C. C. (editor).
Manual of Photogrammetry.
American Society of Photogrammetry, 1980.

[95] Solina, F. and Bajcsy, R.
Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations.
IEEE Transactions on Pattern Analysis and Machine Intelligence 12(2):131-147, February, 1990.

[96] Stark, L. and Bowyer, K.
Generic Recognition Through Qualitative Reasoning About 3-D Shape and Object Function.
In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 251-256. June, 1991.

[97] Suetens, P., Fua, P., and Hanson, A. J.
Computational Strategies for Object Recognition.
Computing Surveys 24(1):5-61, March, 1992.

[98] Tai, A., Kittler, J., Petrou, M., and Windeatt, T.
Vanishing Point Detection.
Image and Vision Computing 11(4):240-245, May, 1993.

[99] Tavakoli, M. and Rosenfeld, A.
Building and Road Extraction from Aerial Photographs.
IEEE Transactions on Systems, Man, and Cybernetics SMC-12(1):84-91, January/February, 1982.

[100] Terzopoulos, D., Witkin, A., and Kass, M.
Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion.
Artificial Intelligence 36:91-123, 1988.

[101] Venkateswar, V. and Chellappa, R.
A Framework for Interpretation of Aerial Images.
In *Proceedings: Tenth International Conference on Pattern Recognition*, pages 204-206. 1990.

[102] Waltz, D.
Understanding Line Drawings of Scenes with Shadows.
In Winston, P. (editor), *The Psychology of Computer Vision*, pages 19-91. McGraw-Hill, New York, 1975.

[103] Weiss, I.
Geometric Invariants and Object Recognition.
International Journal of Computer Vision 10(3):207-231, June, 1993.

[104] Weiss, R. S., Nakatani, H., and Riseman, E. M.
An Error Analysis for Surface Orientation from Vanishing Points.
IEEE Transactions on Pattern Analysis and Machine Intelligence 12(12):1179-1185, December, 1990.

[105] Weyl, H.
Symmetry.
Princeton University Press, Princeton, NJ, 1952.

[106] Willard, D. E.
Predicate-Oriented Database Search Algorithms.
PhD thesis, Harvard University, Aiken Computation Laboratory, Report TR-20-78, 1978.

[107] Witkin, A. P.
Intensity-Based Edge Classification.
In *Proceedings: National Conference on Artificial Intelligence*, pages 36-41. 1982.

[108] Witkin, A. P. and Tenenbaum, J. M.
On the Role of Structure in Vision.
In Beck, J., Hope, B., and Rosenfeld, A. (editors), *Human and Machine Vision*, pages 481-543. Academic Press, New York, 1983.

[109] Zlotnick, A. and Carnine, P. D.
Finding Road Seeds in Aerial Images.
Computer Vision, Graphics and Image Processing: Image Understanding 57(2):243-260, March, 1993.

[110] Zlotnick, A..
Locating Corners in Noisy Curves by Delineating Imperfect Sequences.
Technical Report CMU-CS-88-199, Carnegie Mellon University, December, 1988.

Appendix A
Mathematical Tools

In this appendix, the basic coordinate systems to be used throughout this work are reviewed, as well as the mathematics for manipulating points in image space and object space. This is followed by a discussion of the Gaussian sphere, a finite representation for orientations in 3-space; and vanishing points, points in image space where the projections of parallel object space lines converge. Backprojection and finite image extent bias on the Gaussian sphere are dealt with in separate sections, as are 2D determinant constraints.

Unless otherwise noted, bold face lower case letters denote vectors, and bold face upper case letters denote matrices. $\mathbf{I}$ denotes the identity matrix. $\mathbf{a} \cdot \mathbf{b}$ denotes the dot (scalar, inner) product of $\mathbf{a}$ and $\mathbf{b}$; $\mathbf{a} \times \mathbf{b}$ denotes the cross (vector) product of $\mathbf{a}$ and $\mathbf{b}$. $|\mathbf{a}|$ denotes the magnitude (length) of $\mathbf{a}$. For the sake of brevity, the distinction between row and column vectors will not be strictly enforced; vectors will be written in row form throughout the text. $ab \parallel cd$ indicates that line segment ab is parallel to line segment cd.

The coordinate system and orientation matrix discussions in Section A.1 are based in part on [94]. The descriptions of the Gaussian sphere, vanishing points, and backprojection in Sections A.2, A.3, and A.4 are largely based on Barnard [4]; the only significant exception is that the coordinate systems used in this text are right-handed. Kanatani provides an alternate description of the mathematics of vanishing points in other sources [50, 51]. The description of image extent bias on the Gaussian sphere in Section A.5 is an alternate derivation of earlier results [61].

A.1. Coordinate systems and transformations

The *image coordinate system* is illustrated in Figure A-1. The origin is located in the upper left corner of the image, with the X-axis running vertically, and the Y-axis running horizontally. Rows (values of x) increase from top to bottom, and columns (values of y) increase from left to right.

The *perspective center* is the point at which bundles of perspective rays converge in a lens-camera system. *Fiducial marks* are index marks, fixed relative to the lens cone, which provide reference marks to establish the camera coordinate system and the principal point. The *principal point* is defined as the point on the photographic plane through which a line, passing through the perspective center and perpendicular to the photographic plane, intersects the plane.

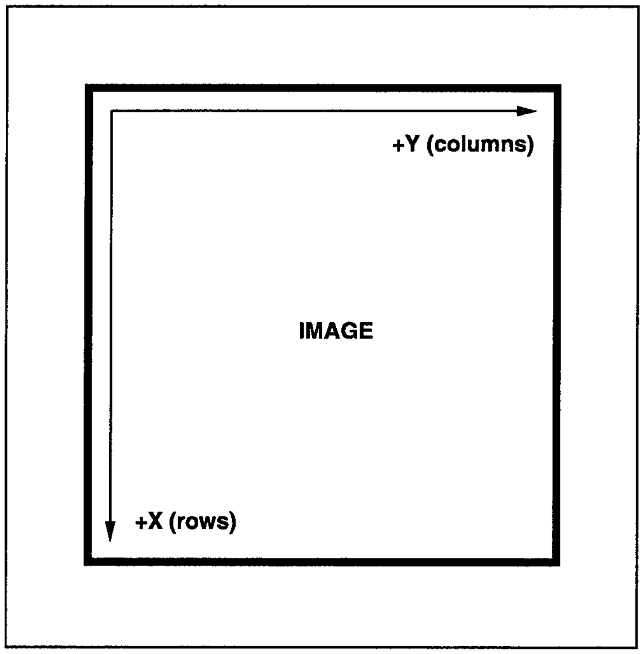


Figure A-1: Image coordinate system

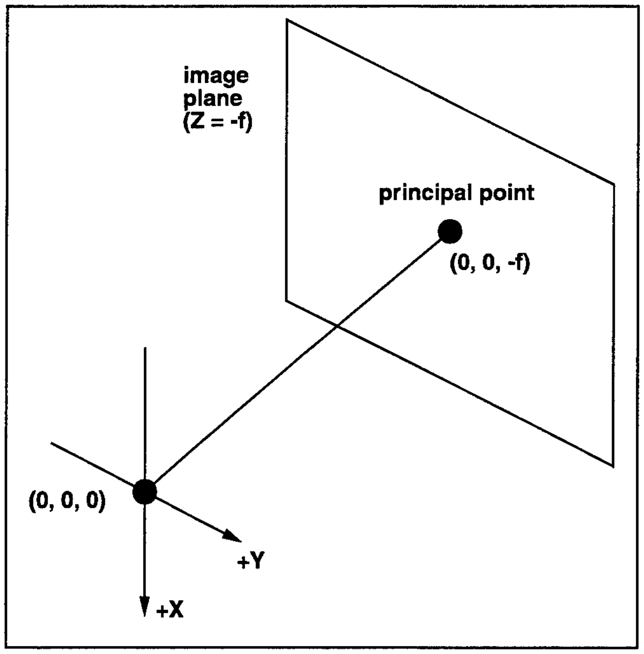


Figure A-2: Camera coordinate system

The *camera coordinate system*, also Cartesian, is illustrated in Figure A-2. The origin is located at the perspective center, with the X and Y axes defined by the fiducial marks on the image (for images which do not have fiducial marks, such as those produced by digital frame cameras, the X and Y axes parallel the image coordinate system X and Y axes). The Z-axis is directed through the principal point of the image. The image plane is defined by $Z = -f$.

The focal length of the camera is denoted by f; it follows that the principal point of the image is located at $(0, 0, -f)$.

The transformation from image coordinates (x_i, y_i) to camera coordinates (x_c, y_c) is typically accomplished by either a 6-parameter affine transform (Equation (A.1)) or an 8-parameter plane projective transform (Equation (A.2)). The parameters of these transformations are determined by a least-squares fit of the fiducial marks measured on the film to those given by the camera calibration report [94].

$$x_c = \frac{a_0 x_i + a_1 y_i + a_2}{x_i + y_i + 1} \qquad y_c = \frac{a_3 x_i + a_4 y_i + a_5}{x_i + y_i + 1} \qquad (A.1)$$

$$x_c = \frac{a_0 x_i + a_1 y_i + a_2}{a_6 x_i + a_7 y_i + 1} \qquad y_c = \frac{a_3 x_i + a_4 y_i + a_5}{a_6 x_i + a_7 y_i + 1} \qquad (A.2)$$

The inverse transformation, from camera to image, can be computed from the direct transformation parameters (Equation (A.3)). In the affine case, $a_6 = a_7 = 1$.

$$d = a_1a_3 - a_0a_4 \qquad b_0 = \frac{a_5a_7 - a_4}{d} \qquad b_1 = \frac{a_1 - a_2a_7}{d} \tag{A.3}$$

$$b_2 = \frac{a_2a_4 - a_1a_5}{d} \qquad b_3 = -\frac{a_5a_6 - a_3}{d} \qquad b_4 = -\frac{a_0 - a_2a_6}{d}$$

$$b_5 = -\frac{a_2a_3 - a_0a_5}{d} \qquad b_6 = \frac{a_4a_6 - a_3a_7}{d} \qquad b_7 = \frac{a_0a_7 - a_1a_6}{d}$$

$$x_i = \frac{b_0x_c + b_1y_c + b_2}{b_6x_c + b_7y_c + 1} \qquad y_i = \frac{b_3x_c + b_4y_c + b_5}{b_6x_c + b_7y_c + 1}$$

The choice of a coordinate system for object space depends upon the intended application. Geodetic, geocentric, and UTM (Universal Transverse Mercator) coordinate systems are used for a variety of tasks in geodesy and photogrammetry. For this work, a natural choice is the *local vertical rectangular coordinate system* [94, p. 484]. This system, also Cartesian, has its origin defined at an arbitrary location on the earth's surface. The *X*-axis points east; the *Y*-axis points north, and the *Z*-axis points vertically. Hereafter, this coordinate system will be referred to as the *world coordinate system.*

A *resection* determines the geographic position of the perspective center and the direction of the optical axis, or the *image orientation.* The image orientation is specified by a 3×3 rotation matrix **M** which rotates the local vertical coordinate system into the camera coordinate system. This matrix is determined by three independent orientation angles, e.g., roll (ω), pitch (ϕ), and yaw (κ), which are computed in the resection (details of resection methods are beyond the scope of this appendix, and can be found elsewhere [94]). Under this parameterization, the entries in the rotation matrix are:

$$\begin{array}{ccc} \cos\phi\cos\kappa & \cos\omega\sin\kappa + \sin\phi\sin\omega\cos\kappa & \sin\omega\sin\kappa - \sin\phi\cos\omega\cos\kappa \\ -\cos\phi\sin\kappa & \cos\omega\cos\kappa - \sin\phi\sin\omega\sin\kappa & \sin\omega\cos\kappa + \sin\phi\cos\omega\sin\kappa \\ \sin\phi & -\cos\phi\sin\omega & \cos\phi\cos\omega \end{array} \tag{A.4}$$

M has the property that

$$\mathbf{M}\mathbf{M}^{\mathrm{T}} = \mathbf{I} \tag{A.5}$$

which can be physically interpreted as representing the fact that a rotation followed by its inverse results in no change in orientation.

The *image tilt* angle τ can be expressed in terms of roll and pitch:

$$\tau = \sin^{-1}\sqrt{\sin^2\phi + \sin^2\omega\cos^2\phi} \tag{A.6}$$

Alternatively, the image tilt angle is also the angle formed by the optical axis with respect to the *Z*-axis of the world coordinate system. The *effective obliquity* angle is a generalization of the tilt angle for specific patches of an image; given some point **p** at the center of an image patch, the effective obliquity angle of the patch is the angle formed by a ray from the perspective center through **p** with the Z-axis of the world coordinate system. When **p** is the principal point, the image tilt angle and the effective obliquity angle are equal.

A.2. The Gaussian sphere

The *Gaussian sphere* is a unit sphere centered at the origin of the camera coordinate system (the perspective center). Any orientation in 3-space can be represented as a point on this sphere by taking a vector representing the orientation and centering it at the origin. The point at which it intersects the surface of the sphere is the Gaussian sphere representation of that orientation. The Gaussian sphere can be used to represent the orientation of either lines (interpreting vectors as direction cosines) or planes (interpreting vectors as planar normals).

Let **n** be a point on the Gaussian sphere. Its position can be expressed in either spherical or Cartesian coordinates. Let α be the *azimuth* of **n**, the angle measured from the Z-axis in the YZ-plane. Let β be the *elevation* of **n**, the angle measured from the Z-axis in the XZ-plane. The Cartesian coordinates can then be expressed in vector form as

$$\mathbf{n} = [\sin\beta, \sin\alpha\cos\beta, \cos\alpha\cos\beta] \tag{A.7}$$

Each line segment in an image formed by central projection is associated with a plane in the camera coordinate system, known as its *interpretation plane*. Let $\mathbf{p}_1 = [x_1, y_1, -f]$ and $\mathbf{p}_2 = [x_2, y_2, -f]$ be two image points defining a line L. The interpretation plane of L is then defined by three points: $\mathbf{p}_1$, $\mathbf{p}_2$, and the origin (the perspective center). It can be represented by the unit normal vector:

$$\phi = \frac{\mathbf{p}_1 \times \mathbf{p}_2}{|\mathbf{p}_1||\mathbf{p}_2|} \tag{A.8}$$

The interpretation plane is so named because the line in space which projects to the imaged line segment L must lie in ϕ. ϕ can be said to represent the space of possible locations, or interpretations, of the world line corresponding to L.

Since, by definition, each interpretation plane passes through the origin, the intersection of each interpretation plane with the Gaussian sphere forms a great circle on the sphere. Let **g** be any point on the great circle, expressed in vector form. Since ϕ, the normal of the interpretation plane, is orthogonal to **g**, the equation of the great circle is

$$\mathbf{g} \cdot \phi = 0 \tag{A.9}$$

Combining this equation with Equation (A.7), we can derive an expression for elevation in terms of the azimuth and the interpretation plane unit normal:

$$\beta(\alpha, \phi) = \tan^{-1} \frac{-\phi_y \sin\alpha - \phi_z \cos\alpha}{\phi_x} \tag{A.10}$$

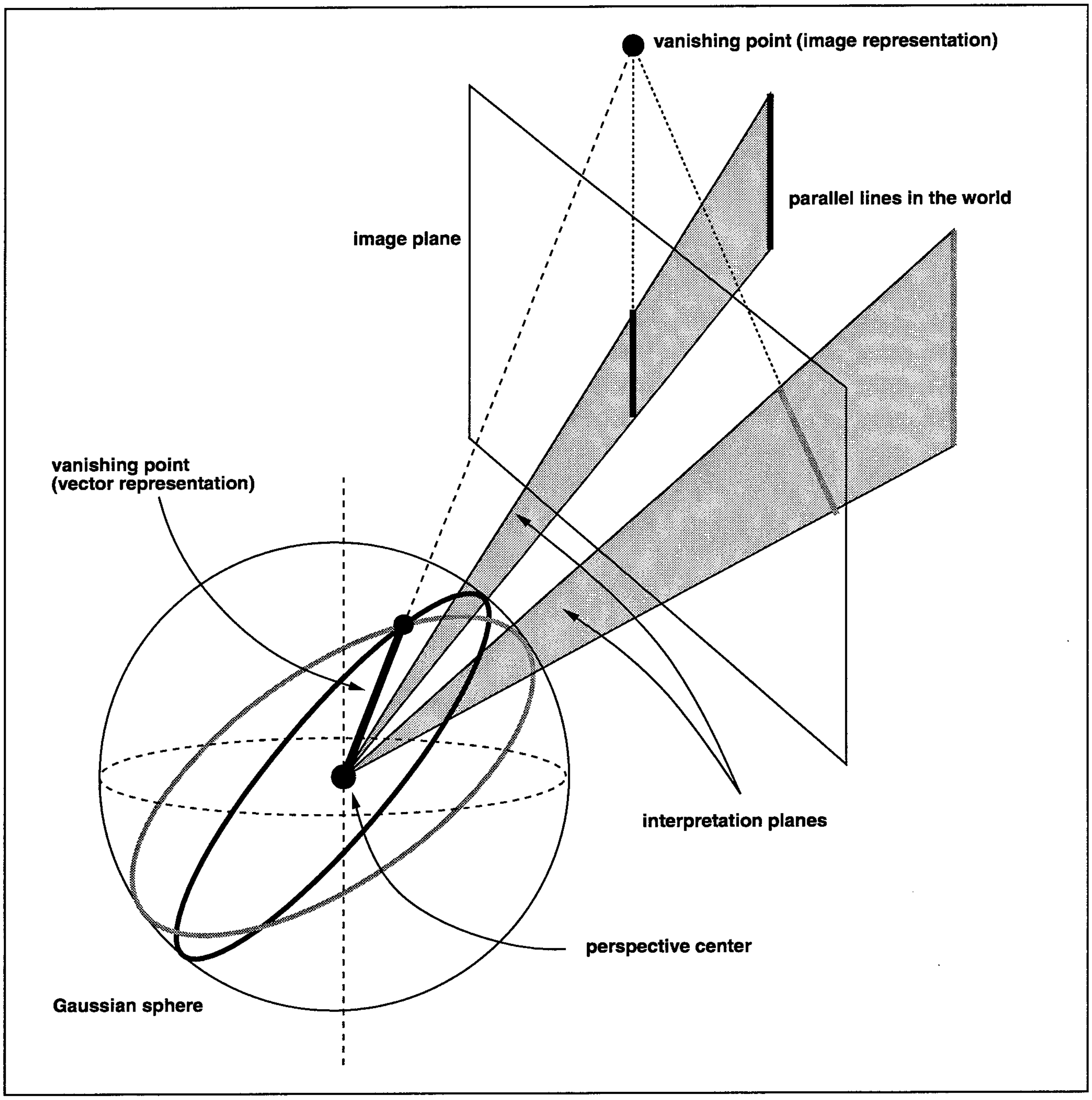


Figure A-3: Vanishing points, interpretation planes, and the Gaussian sphere

When ϕ_x is small, an alternative form can be derived which expresses azimuth in terms of elevation and the interpretation plane unit normal:

$$\alpha(\beta, \phi) = \tan^{-1} \frac{-\phi_z}{\phi_y} \qquad (A.11)$$

A.3. Vanishing points

Under central projection, a set of parallel lines in the world maps to a set of lines in image space which converge on a single point, known as the *vanishing point*. A vanishing point can be represented in two ways: as a point on the image plane ($\mathbf{p} = <x, y, -f>$) or as a point on the Gaussian sphere (alternatively, a unit vector placed at the perspective center):

$$\mathbf{v} = \frac{\mathbf{p}}{|\mathbf{p}|} \tag{A.12}$$

Figure A-3 depicts the geometric situation. Two parallel lines in the world, colored black and grey, project to two convergent line segments in the image plane. The intersection of the extended line segments is the vanishing point's image representation, as a point on the image plane. Each of the line segments, together with the perspective center, forms an interpretation plane. These interpretation planes generate great circles on the sphere, also colored black and grey to indicate the lines which generated them.

The great circles intersect at two points on the Gaussian sphere, which can be regarded as two unit vectors placed at the perspective center. We discard the vector pointing away from the image plane; the remaining vector represents the vanishing point. The vector and image representations of the vanishing point are related; the vanishing point vector pierces the image plane at precisely the coordinates of the image representation of the vanishing point. This relationship allows us to convert between representations readily (assuming the vector is not parallel to the image plane, in which case the image representation is not well-defined).

Given a vector $\mathbf{q}$ in the world coordinate system, its vanishing point on the Gaussian sphere can be readily computed by applying the image orientation matrix:

$$\mathbf{v} = \frac{\mathbf{Mq}}{|\mathbf{Mq}|} \tag{A.13}$$

This permits direct computation of the vanishing point of vertical lines in the world by setting

$$\mathbf{q} = [0,0,1]^T \tag{A.14}$$

and applying Equation (A.13). Similarly, if the solar azimuth θ is known, then setting

$$\mathbf{q} = [\sin\theta, \cos\theta, 0]^T \tag{A.15}$$

and applying Equation (A.13) produces the vanishing point for the shadow lines cast on horizontal planes by vertical lines in object space. While these two object space orientations are of particular interest, it is important to realize that any orientation in the world can be translated to a point on the Gaussian sphere by Equation (A.13).

Let $\mathbf{S}$ be a set of parallel lines in object space, sharing the unit orientation vector $\mathbf{u}$. The normals of the interpretation planes of the lines in $\mathbf{S}$ form a great circle on the Gaussian sphere, which itself defines a plane. The unit normal $\mathbf{v}$ of this plane is the result of applying Equation (A.13) to $\mathbf{u}$:

$$\mathbf{v} = \frac{\mathbf{Mu}}{|\mathbf{Mu}|} \qquad (A.16)$$

A.4. Backprojection

Geometric properties such as the intersection of lines in projective space are *descriptive* properties: they are preserved under projection. However, *metric* properties such as angles and curvatures are not preserved under projection. *Backprojection* is a general method for exploiting metric features to determine planar orientations, by essentially inverting the projective transform and selecting the most feasible inverse projections of image features. This section briefly addresses the basic mathematics involved in backprojection.

Consider the geometric situation in Figure A-4. A plane is tangent to the Gaussian sphere at a point γ with azimuth α and elevation β. This plane can be represented by its unit normal, which is given by Equation (A.7). The plane equation can then be expressed as:

$$\gamma(\alpha,\beta)\cdot[x,y,z] = 1 \qquad (A.17)$$

Planar orientation can be computed by measuring image features projected onto γ for all values of azimuth and elevation, and selecting the measurements which satisfy some constraint or heuristic. For brevity, one specific case is considered here: backprojection of angle magnitudes.

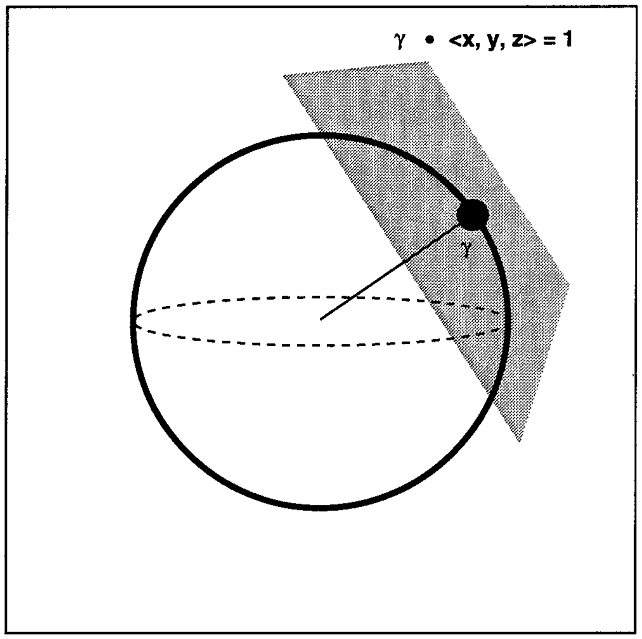


Figure A-4: Backprojection geometry

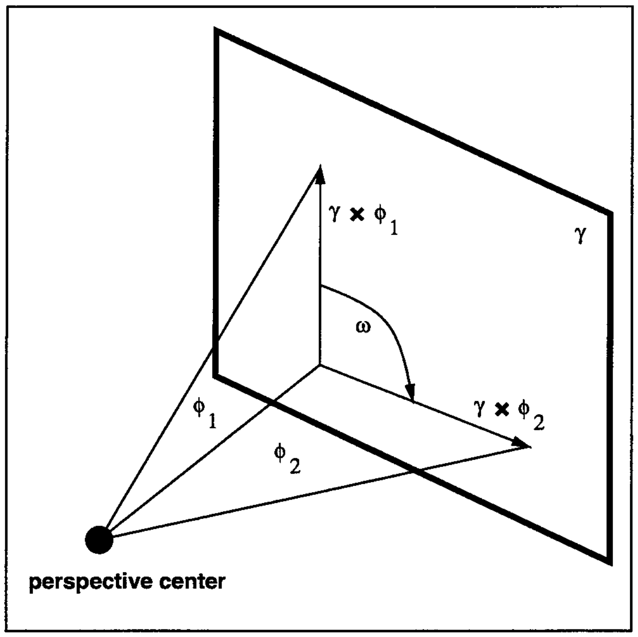


Figure A-5: Backprojection of angles

Consider the geometric situation in Figure A-5. Two interpretation planes ϕ_1 and ϕ_2 form an angle ω on the backprojection plane γ. Since the dot product of two vectors is the product of their magnitudes and the cosine of the angle they form, the following equation holds:

$$\cos\omega = \frac{(\gamma \times \phi_1) \cdot (\gamma \times \phi_2)}{|\gamma \times \phi_1|\,|\gamma \times \phi_2|} \qquad (A.18)$$

Using Equation (A.18), each angle (image feature) can be projected onto planes of all orientations (represented as points on the Gaussian sphere), and the resulting value at each point is the angle the image feature must make on the plane represented by that point. Barnard [4] and Horaud [38] discuss angle backprojection in detail; Horaud also considers the angular constraints formed by the junction of three lines.

A.5. Finite image extent bias

As a result of the finite extent of the image, the probability of detecting a vanishing point anywhere on the Gaussian sphere is non-uniform [15, 64]. In particular, all great circles will produce votes inside the projection of the image boundary onto the sphere. This implies that the likelihood of great circle intersection is higher inside this boundary than it is elsewhere on the sphere.

A technique developed to address this bias [61] involves generating a probability mask with the same dimensions as the Gaussian sphere histogram. This probability mask is then used to weight the values in the histogram. The original formulation of this solution was phrased in terms of integral calculus; the solution derived here is expressed in terms of simple vector algebra. For simplicity, we consider only the half-sphere which faces the image plane, since every vanishing point is represented twice on the sphere; once on the side facing the image plane, and once on the other side of the sphere, where the line through the first point and the perspective center intersects the far side of the sphere.

Consider a point **u** on the Gaussian half-sphere, and every possible great circle that passes through **u**. This set of great circles sweeps out some area *A* on the half-sphere. The probability of detecting the vanishing point **u** is proportional to a fractional quantity: the ratio of *A* to the total area of the half-sphere. So, the goal is to compute this ratio for every point on the sphere, given an image (the extent and position of which necessarily defines the set of great circles which are possible for any point on the sphere).

Figures A-6 and A-7 depict the geometric situation. Figure A-6 shows the great circles created by the projection of the lines bordering the image onto the Gaussian sphere. This diagram is rendered from inside the sphere, looking out towards the image plane. Solid lines are either in or on the sphere; dotted lines are outside the sphere. The large dot is the perspective center; the diagonal lines are the lines of projection of the image corners. The image border *top* projects to create the half-circle T; *bottom* projects to B, *left* projects to L, and *right* projects to R. **UL**, **UR**, **LL**, and **LR** denote the points where the corner projections intersect the sphere (and hence they also represent vectors placed at the perspective center).

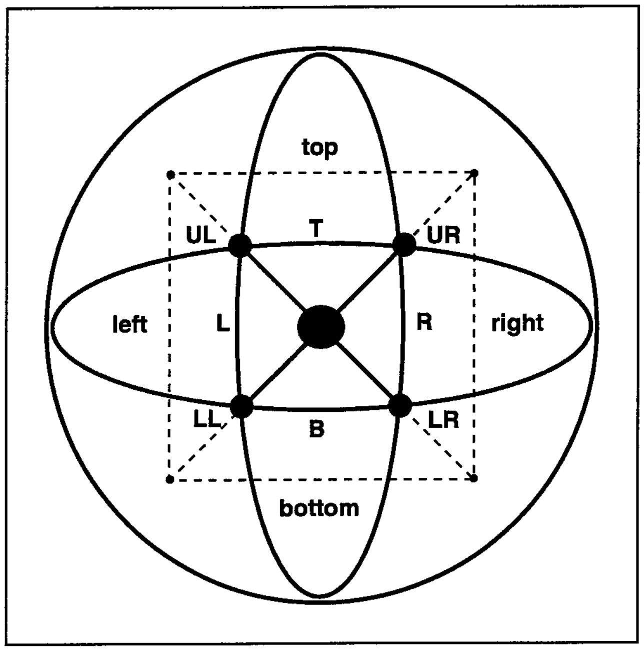


Figure A-6: Image projection on Gaussian sphere

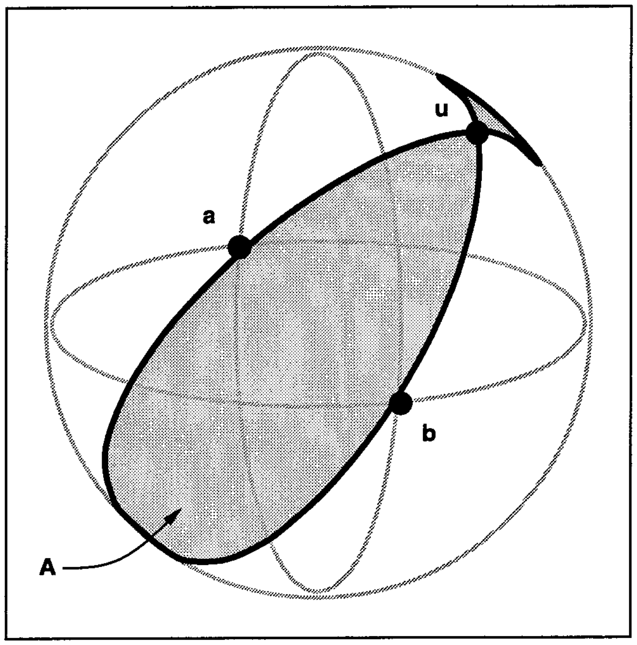


Figure A-7: Area covered by great circles

Figure A-7 shows a point **u** on the Gaussian half-sphere. **u** lies outside the projection of the image boundary on the sphere. Since any great circle passing through **u** must pass through the image, **a** and **b** bound the area A, shaded in the diagram, in which any such great circle must lie. Note that a point which lies inside the projection of the image boundary on the sphere generates an area which covers the entire sphere, since every great circle passing through the point also lies in the image.

If **u** lies inside the projection of the image on the sphere, let $P(\mathbf{u})=\pi$. The motivation for this definition will become clear momentarily. If **u** lies outside the projection of the image on the sphere, then let **a** and **b** be the vectors, placed at the perspective center, which pierce the extreme corners of the image relative to **u**. The great circles defined by **u** and each of **a** and **b** bound an area A. This is the situation depicted in Figure A-7, as described earlier.

To compute the ratio of the area of A to the area of the half-sphere, it suffices to measure the angle between the two great circles, since this angle is proportional to the surface area of A. Each great circle lies in a plane passing through the perspective center; then $\mathbf{a} \times \mathbf{u}$ and $\mathbf{b} \times \mathbf{u}$ are the normals of these planes. The vector angle of these normals is then:

$$\theta = \cos^{-1} \frac{(\mathbf{a} \times \mathbf{u}) \cdot (\mathbf{b} \times \mathbf{u})}{|\mathbf{a} \times \mathbf{u}|\,|\mathbf{b} \times \mathbf{u}|} \qquad (A.19)$$

Hence, when **u** lies outside the projection of the image on the sphere, we set $P(\mathbf{u})=\theta$. Since the vector angle ranges from 0 to π, this explains the use of $P(\mathbf{u})=\pi$ when **u** lies inside the image projection on the sphere, since the entire half-sphere is covered in this case.

It remains to calculate the corner points which define the extreme bounds of A, given a point **u** which lies outside of the image projection on the sphere. Each image border generates a great circle on the sphere, and hence forms a plane passing through the perspective center. The unit normals of each plane can be computed directly:

$$\mathbf{top} = \frac{\mathbf{UL} \times \mathbf{UR}}{|\mathbf{UL} \times \mathbf{UR}|} \qquad \mathbf{right} = \frac{\mathbf{UR} \times \mathbf{LR}}{|\mathbf{UR} \times \mathbf{LR}|} \tag{A.20}$$

$$\mathbf{bottom} = \frac{\mathbf{LR} \times \mathbf{LL}}{|\mathbf{LR} \times \mathbf{LL}|} \qquad \mathbf{left} = \frac{\mathbf{LL} \times \mathbf{UL}}{|\mathbf{LL} \times \mathbf{UL}|}$$

These vectors all point away from the image projection on the sphere. Hence, four dot product tests will determine where **u** lies, and this determines which two corner points define the extreme bounds of the area A. The dot product tests are as follows:

$$\text{if } (\mathbf{top} \cdot \mathbf{u}) > 0 \quad \mathbf{u} \text{ is above } \mathbf{T} \tag{A.21}$$

$$\text{if } (\mathbf{right} \cdot \mathbf{u}) > 0 \quad \mathbf{u} \text{ is right of } \mathbf{R}$$

$$\text{if } (\mathbf{bottom} \cdot \mathbf{u}) > 0 \quad \mathbf{u} \text{ is below } \mathbf{B}$$

$$\text{if } (\mathbf{left} \cdot \mathbf{u}) > 0 \quad \mathbf{u} \text{ is left of } \mathbf{L}$$

Once the probability mask has been computed, the histogram can be scaled accordingly. A natural approach, given an azimuth-elevation quantization of the Gaussian sphere S, is to compute the following:

$$S_{ij} := \frac{S_{ij}}{P_{ij}}, \forall i \forall j \tag{A.22}$$

A.6. 2D determinant tests

The 2D determinant is a basic tool for testing convexity and half-plane membership in two dimensions. To begin the discussion, consider the line equation in determinant form, where (x_1, y_1) and (x_2, y_2) are two points on the line:

$$x(y_1 - y_2) + y(x_1 - x_2) + x_1 y_2 - x_2 y_1 = 0 \tag{A.23}$$

Let a, b, and c be three points in the plane. Let **v** be the 2D vector originating at b and terminating at c. **v** generates a line which divides the plane into two half-planes. We can use Equation (A.23) directly to compute a quantity γ suitable for testing the half-plane membership of a:

$$\gamma(a,b,c) = a_x(b_y - c_y) + a_y(b_x - c_x) + b_x c_y - c_x b_y \tag{A.24}$$

If $\gamma > 0$, then a lies to the left of **v** when **v** is oriented to point upward. If $\gamma < 0$, then a lies to the right of **v**. If $\gamma = 0$, then a lies on the line formed by **v**. The test for corner rotation direction is essentially identical. Given a corner formed by the points b, c, and a, in that order, then γ gives the rotation. If $\gamma > 0$, then the corner rotates counterclockwise. If $\gamma < 0$, then the corner rotates clockwise. If $\gamma = 0$, then the corner is actually a straight line through b, c, and a.

The convexity of a 2-corner (described in Section 4.3) can be readily computed with Equation (A.24). Consider a 2-corner with vertices *a*, *b*, *c*, and *d*, and assume that no three consecutive points of the 2-corner lie on a line. Since the rotation at corners *abc* and *bcd* must be the same, and hence have the same sign according to γ, then the 2-corner is convex if and only if:

$$\gamma(c,a,b)\,\gamma(d,b,c) \;>\; 0 \tag{A.25}$$

As noted in Section 4.3, 2-corners which are expected to form triangular facets require additional geometric tests to ensure their validity. In particular, a 2-corner can be convex according to Equation (A.25), yet the intersection point of its arms and the arms themselves can lie in opposite half-planes. In this case, the 2-corner cannot possibly form a triangle, and hence this interpretation can be discarded.

Consider a 2-corner with vertices *a*, *b*, *c*, and *d*. Let *x* be the intersection point of the two lines formed by the arms of the 2-corner, the line segments *ab* and *cd*. If $ab \parallel cd$, then clearly no triangle is possible. Assuming the 2-corner has passed the convexity test, then it suffices to test whether *a* and *x* both lie in the same half-plane produced by the central edge *bc*:

$$\gamma(a,b,c)\,\gamma(x,b,c) \;>\; 0 \tag{A.26}$$

Appendix B
Experimental Results

This appendix presents the results of the comparative building extraction experiments described in Chapter 6. First, four tables are presented which describe the test imagery in detail; then, the actual results of the four building extraction systems on all of the test images are presented, along with performance measures.

The test data used in the comparative experiments of Chapter 6 consists of 83 images covering 18 distinct sites. Each scene occurs at least twice in the test imagery, covered from at least two different viewpoints. Image acquisition details for each of the test images are summarized in Tables B-1 and B-2. The first table gives information for imagery of Fort Hood, Texas; this site has been a focal point for the RADIUS research program, and a significant amount of imagery from a wide range of viewpoints has been made available for the site. The second table gives information for imagery collected for other geographic areas.

Tables B-3 and B-4 give specific measurements of various image and ground-truth related quantities used in the analysis of Chapter 6. Most of these quantities are described thoroughly in that chapter. The exception is GSD (ground sample distance); since many of the test images exhibit significant obliquity, the ground sample distance can vary across a particular image. To obtain GSDs for each image, the object space area covered by each image was computed, and divided by the number of pixels in the image. This gives the average object space area covered by a single pixel, and the square root of this is the GSD measurement presented in the tables.

Following these tables, the building extraction results for BUILD, BUILD+SHAVE, VHBUILD, and PIVOT are presented for each of the 83 test images. Note that the results for BUILD and BUILD+SHAVE are condensed into one graphic for each image; BUILD+SHAVE simply takes the rooftop polygons produced by BUILD, and adds wall polygons.

The images are presented in alphabetical order, one per page, with four figures on each page. The first figure on each page shows the raw image with no results, to give the reader a clear view of the image. The other three figures show the results for BUILD+SHAVE, VHBUILD, and PIVOT. A table at the bottom of each page presents the 2D and 3D performance statistics for each system on that image. Boldface numbers indicate the best score for a particular metric across all four building extraction systems.

scene	images	scene description	source
COMPLEX	COMPLEX_FHN713 COMPLEX_FHN715 COMPLEX_FHN717 COMPLEX_FHOV1627 COMPLEX_FHOV1727 COMPLEX_FHOV525 COMPLEX_FHOV727 COMPLEX_FHOV927	multi-level rectilinear structure	distributed by the RADIUS project; digitized and resected at Digital Mapping Laboratory, CMU
DARK	DARK_FHN717 DARK_FHN719 DARK_FHOV1627 DARK_FHOV1727 DARK_FHOV425 DARK_FHOV527 DARK_FHOV727	split-level peaked roof buildings	distributed by the RADIUS project; digitized and resected at Digital Mapping Laboratory, CMU
NEST	NEST_FHN711 NEST_FHN78 NEST_FHOV1027 NEST_FHOV525 NEST_FHOV625 NEST_FHOV927	one complex structure with multi-level rectilinear wings and an overhanging curved ledge; several peaked roof buildings with overhangs	distributed by the RADIUS project; digitized and resected by Digital Mapping Laboratory, CMU
RADT10	RADT10 RADT10OB RADT10S RADT10WOB	flat roof rectilinear buildings and peaked roof buildings	distributed by the RADIUS project; resected by Digital Mapping Laboratory, CMU
RADT11	RADT11 RADT11OB RADT11S RADT11WOB	flat roof rectilinear buildings and peaked roof buildings	distributed by the RADIUS project; resected by Digital Mapping Laboratory, CMU
RADT5	RADT5 RADT5OB RADT5S RADT5WOB	flat roof rectilinear buildings and peaked roof buildings	distributed by the RADIUS project; resected by Digital Mapping Laboratory, CMU
RADT6	RADT6 RADT6OB RADT6S RADT6WOB	flat roof rectilinear buildings and peaked roof buildings	distributed by the RADIUS project; resected by Digital Mapping Laboratory, CMU
RADT9	RADT9 RADT9OB RADT9S RADT9WOB	flat roof rectilinear buildings and peaked roof buildings	distributed by the RADIUS project; resected by Digital Mapping Laboratory, CMU
T	T_FHN711 T_FHN78 T_FHOV1027 T_FHOV525 T_FHOV625 T_FHOV927	composite flat roof buildings with gabled perimeters	distributed by the RADIUS project; digitized and resected by Digital Mapping Laboratory, CMU
VANILLA	VANILLA_FHN711 VANILLA_FHN713 VANILLA_FHOV1627 VANILLA_FHOV525 VANILLA_FHOV625 VANILLA_FHOV927	several peaked roof buildings, some with overhangs	distributed by the RADIUS project; digitized and resected by Digital Mapping Laboratory, CMU

Table B-1: Summary of test imagery over Fort Hood, Texas

scene	geographic location	images	scene description	source
AVENCHINDUST	Avenches, Switzerland	AVENCHINDUST1_4 AVENCHINDUST2_4 AVENCHINDUST3_4 AVENCHINDUST4_4	industrial area comprised of peaked roof buildings with overhangs and sloped buildings	digitized and resected by ETH, Zurich, for 1995 Ascona, Switzerland workshop on building extraction; new resections done at Digital Mapping Laboratory, CMU
BLOCK	Riverside, California	BLOCK_RI0707 BLOCK_RI0808	rectilinear flat roof structures in a city block	digitized and resected by Analytical Surveys, Inc. (ASI)
FLAT	Finland	FLAT_L FLAT_R	peaked roof buildings with overhangs and gables	distributed as part of an ISPRS test of image understanding techniques [28]; new resections done at Digital Mapping Laboratory, CMU
J	modelboard; no geographic location	J1WHOLE J2TRIM J3WHOLE J4TRIM J6WHOLE J7WHOLE J24WHOLE	images of a modelboard with simple, composite and complex rectilinear building models with superstructure	acquired with digital camera for RADIUS project; resected at Digital Mapping Laboratory, CMU
MIXT	Pittsburgh, Pennsylvania	MIXT_SLAG15 MIXT_SLAG16 MIXT_SLAG25 MIXT_SLAG26	simple and complex rectilinear buildings, some with gabled roof perimeters	digitized and resected by USATEC for DARPA U/ALV project
MMBLD	Denver, Colorado	MMBLD-70OV MMBLD-71OV MMBLD-8OV	one multi-level rectilinear structure	Digitized and resected by USATEC
ORCHARD	Riverside, California	ORCHARD_RI0607 ORCHARD_RI0608 ORCHARD_RI0609 ORCHARD_RI0707 ORCHARD_RI0708 ORCHARD_RI0709	rectangular and peaked-roof barns near an orchard	digitized and resected by Analytical Surveys, Inc. (ASI)
VILLAGE	Fort Benning, Georgia	VILLAGE_FTBENN407 VILLAGE_FTBENN408	military training site with flat roof and peaked roof buildings, some roofs with superstructure	digitized and resected by USATEC for DMSO Rapid Construction of Virtual Worlds (RCVW) program

Table B-2: Summary of test imagery

scene	images	GSD (m)	image dimensions (# rows × # columns)	image tilt angle (degrees)	effective obliquity angle (degrees)	average object complexity (elements/bld)	object space density (elements/ 1000m2)	image space density (elements/ 1000 pixels)
COMPLEX	COMPLEX_FHN713	0.32	556 × 642	0.5	31.8	112	6.0	0.63
	COMPLEX_FHN715	0.32	540 × 628	0.6	10.2	112	6.3	0.66
	COMPLEX_FHN717	0.33	541 × 632	0.5	27.6	112	6.2	0.66
	COMPLEX_FHOV1627	0.58	290 × 378	18.6	33.8	112	6.2	2.04
	COMPLEX_FHOV1727	0.59	278 × 405	19.8	38.9	92	7.0	2.45
	COMPLEX_FHOV525	0.47	387 × 535	34.6	31.1	104	6.8	1.51
	COMPLEX_FHOV727	0.46	403 × 443	28.2	20.8	112	6.0	1.25
	COMPLEX_FHOV927	0.62	319 × 395	24.8	45.7	104	6.4	2.48
DARK	DARK_FHN717	0.32	1032 × 912	0.5	21.9	54	7.7	0.80
	DARK_FHN719	0.32	1042 × 914	0.4	28.2	54	7.6	0.79
	DARK_FHOV1627	0.48	693 × 714	18.6	34.3	54	6.5	1.53
	DARK_FHOV1727	0.48	691 × 634	19.8	16.5	54	7.3	1.73
	DARK_FHOV425	0.63	489 × 562	36.1	36.4	54	6.9	2.75
	DARK_FHOV527	0.90	262 × 508	49.2	50.7	54	7.0	5.68
	DARK_FHOV727	0.58	532 × 595	28.2	32.2	54	7.0	2.39
NEST	NEST_FHN711	0.32	527 × 488	0.7	25.2	80.5	47.8	5.01
	NEST_FHN78	0.32	532 × 491	1.1	21.5	80.5	47.4	4.93
	NEST_FHOV1027	0.49	376 × 303	20.2	28.8	80.5	47.2	11.31
	NEST_FHOV525	0.47	360 × 448	34.6	35.1	80.5	36.1	7.99
	NEST_FHOV625	0.43	423 × 338	42.5	18.1	80.5	47.6	9.01
	NEST_FHOV927	0.45	364 × 379	24.8	19.9	80.5	46.2	9.34
RADT10	RADT10	0.43	480 × 512	3.2	20.3	43.1	8.4	1.58
	RADT10OB	0.41	834 × 483	29.4	42.3	41.8	6.7	1.14
	RADT10S	0.44	515 × 547	2.5	24.8	42.4	7.9	1.51
	RADT10WOB	0.53	547 × 348	30.3	53.3	42.4	7.8	2.23
RADT11	RADT11	0.44	480 × 512	3.1	20.9	43.3	2.7	0.53
	RADT11OB	0.44	670 × 633	29.4	42.7	43.3	1.6	0.31
	RADT11S	0.44	498 × 540	2.6	11.6	43.3	2.5	0.48
	RADT11WOB	0.48	660 × 416	30.3	49.6	43.3	2.0	0.47
RADT5	RADT5	0.43	512 × 512	3.2	26.5	35.4	12.3	2.30
	RADT5OB	0.48	571 × 506	29.3	48.5	35.5	11.8	2.71
	RADT5S	0.43	550 × 559	2.6	26.8	35.5	11.7	2.19
	RADT5WOB	0.59	533 × 293	30.3	56.5	35.4	11.2	3.85
RADT6	RADT6	0.44	512 × 512	3.2	9.7	74.5	5.8	1.14
	RADT6OB	0.37	639 × 751	29.4	32.9	74.5	4.6	0.62
	RADT6S	0.44	537 × 537	2.7	17.9	74.5	5.3	1.03
	RADT6WOB	0.45	639 × 496	30.2	45.1	74.5	4.7	0.94
RADT9	RADT9	0.43	743 × 759	2.9	30.0	35.8	6.8	1.27
	RADT9OB	0.53	463 × 506	29.1	51.9	36.4	7.8	2.18
	RADT9S	0.43	757 × 825	2.2	20.5	37.1	6.4	1.19
	RADT9WOB	0.55	852 × 494	30.3	55.4	37.1	5.8	1.76
T	T_FHN711	0.33	867 × 828	0.7	23.1	43.4	9.1	0.97
	T_FHN78	0.32	871 × 833	1.1	22.5	43.4	9.2	0.96
	T_FHOV1027	0.53	561 × 518	20.2	31.4	45.1	8.9	2.48
	T_FHOV525	0.55	487 × 678	34.6	38.3	52.4	8.4	2.54
	T_FHOV625	0.52	578 × 531	42.5	26.8	43.4	8.3	2.26
	T_FHOV927	0.50	522 × 593	24.8	27.7	43.4	9.0	2.25
VANILLA	VANILLA_FHN711	0.32	735 × 879	0.7	9.9	43.6	25.4	2.63
	VANILLA_FHN713	0.32	733 × 870	0.5	21.7	43.6	25.4	2.67
	VANILLA_FHOV1627	0.55	407 × 590	18.6	37.2	44.1	24.9	7.53
	VANILLA_FHOV525	0.50	459 × 610	34.6	24.5	43.6	24.1	6.08
	VANILLA_FHOV625	0.49	539 × 744	42.5	32.3	44.3	20.0	4.75
	VANILLA_FHOV927	0.53	446 × 571	24.8	25.9	44.3	24.5	6.96

Table B-3: Data for test imagery over Fort Hood, Texas

scene	images	GSD (*m*)	image dimensions (# rows × # columns)	image tilt angle (degrees)	effective obliquity angle (degrees)	average object complexity (elements/bld)	object space density (elements/ 1000*m*2)	image space density (elements/ 1000 pixels)
AVENCHINDUST	AVENCHINDUST1_4	0.29	450 × 450	1.0	26.4	77.2	22.5	1.91
	AVENCHINDUST2_4	0.29	450 × 450	1.0	30.8	77.2	22.1	1.91
	AVENCHINDUST3_4	0.29	450 × 450	1.8	14.4	77.2	22.1	1.91
	AVENCHINDUST4_4	0.29	450 × 450	0.5	20.1	77.2	22.4	1.91
BLOCK	BLOCK_RI0707	0.35	594 × 781	3.7	21.1	48.4	13.0	1.56
	BLOCK_RI0808	0.32	666 × 837	4.1	19.9	48.4	12.6	1.30
FLAT	FLAT_L	0.24	1000 × 1000	0.7	25.6	48.2	13.7	0.82
	FLAT_R	0.24	1000 × 1000	0.4	6.2	48.2	13.8	0.82
J	J1WHOLE	0.58	1039 × 1306	32.4	32.4	51.5	6.0	2.01
	J2TRIM	0.62	1037 × 1053	42.2	42.2	53.4	6.1	2.30
	J3WHOLE	0.53	1041 × 1312	2.9	2.9	53.4	6.0	1.68
	J4TRIM	0.57	1041 × 1255	29.6	29.9	53.6	6.3	2.05
	J6WHOLE	0.54	1038 × 1308	14.9	14.9	53.2	6.5	1.88
	J7WHOLE	0.54	1043 × 1313	15.3	15.4	55.8	6.4	1.88
	J24WHOLE	0.54	1039 × 1310	13.8	13.8	52.8	6.7	1.98
MIXT	MIXT_SLAG15	0.61	423 × 499	0.6	28.4	44.6	24.6	9.08
	MIXT_SLAG16	0.61	417 × 476	0.7	30.1	44.6	26.0	9.65
	MIXT_SLAG25	0.60	447 × 514	0.2	27.6	44.6	22.9	8.34
	MIXT_SLAG26	0.60	454 × 502	0.5	32.2	44.6	23.0	8.41
MMBLD	MMBLD-70OV	0.51	334 × 258	12.3	12.0	230	10.2	2.67
	MMBLD-71OV	0.52	322 × 262	19.5	15.9	230	10.1	2.73
	MMBLD-8OV	0.36	426 × 429	26.1	27.2	230	9.8	1.26
ORCHARD	ORCHARD_RI0607	0.36	472 × 501	4.1	28.5	39	12.2	1.65
	ORCHARD_RI0608	0.35	475 × 506	4.1	16.5	39	12.9	1.62
	ORCHARD_RI0609	0.34	509 × 526	3.4	32.5	39	12.7	1.46
	ORCHARD_RI0707	0.32	530 × 557	3.7	33.9	39	12.5	1.32
	ORCHARD_RI0708	0.34	494 × 534	3.9	16.8	39	12.8	1.48
	ORCHARD_RI0709	0.36	459 × 509	4.6	31.5	39	12.7	1.67
VILLAGE	VILLAGE_FTBENN407	0.17	858 × 712	0.8	10.5	60.8	78.8	2.39
	VILLAGE_FTBENN408	0.17	859 × 708	1.1	22.2	60.8	79.8	2.40

Table B-4: Data for test imagery

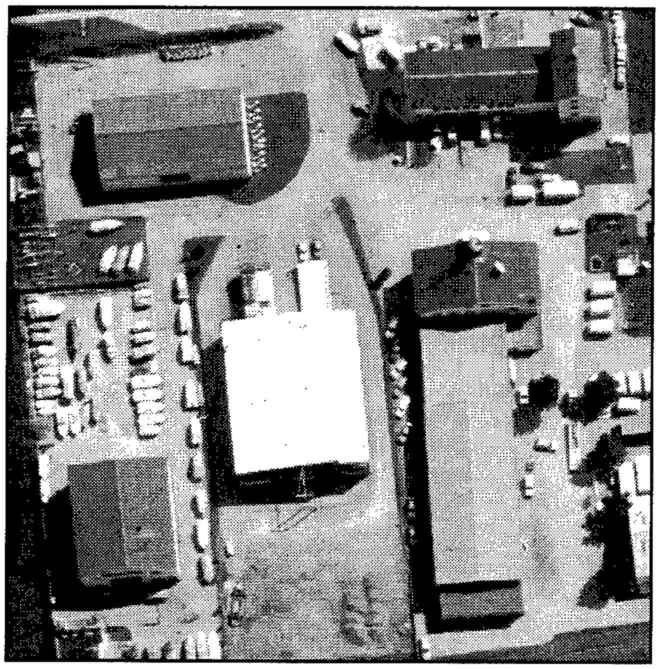


Figure B-1: AVENCHINDUST1_4 image

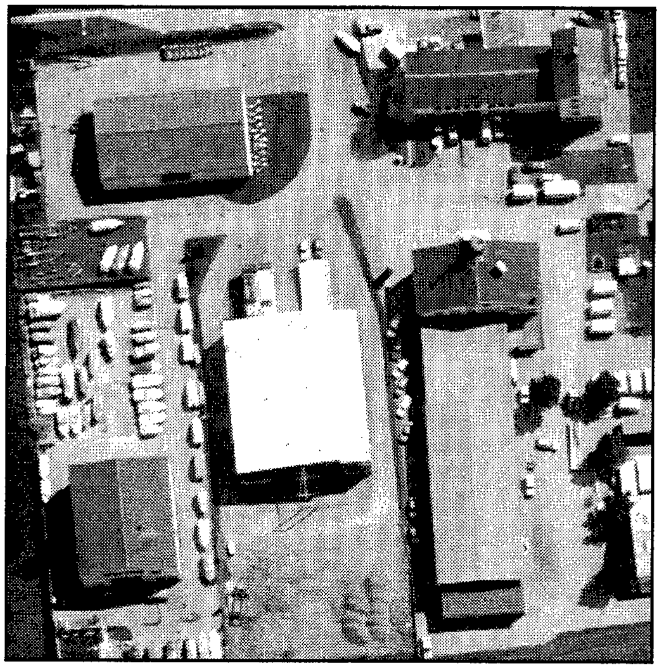


Figure B-2: BUILD+SHAVE results, AVENCHINDUST1_4

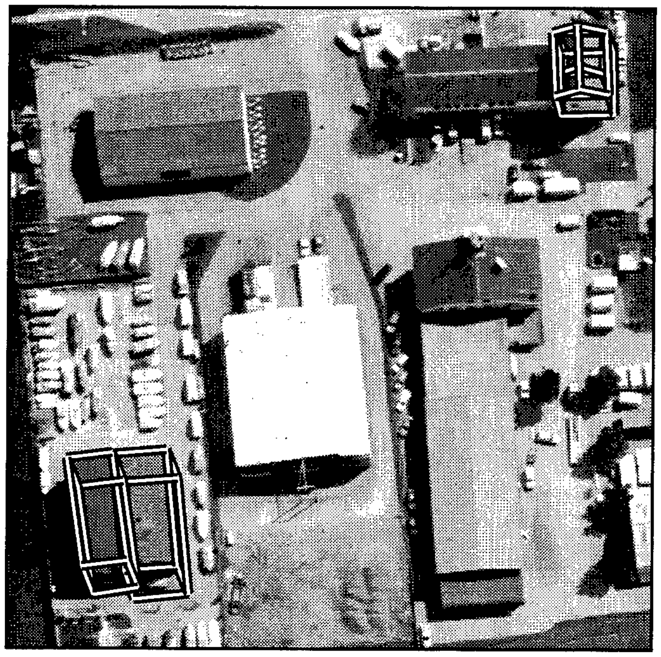


Figure B-3: VHBUILD results, AVENCHINDUST1_4

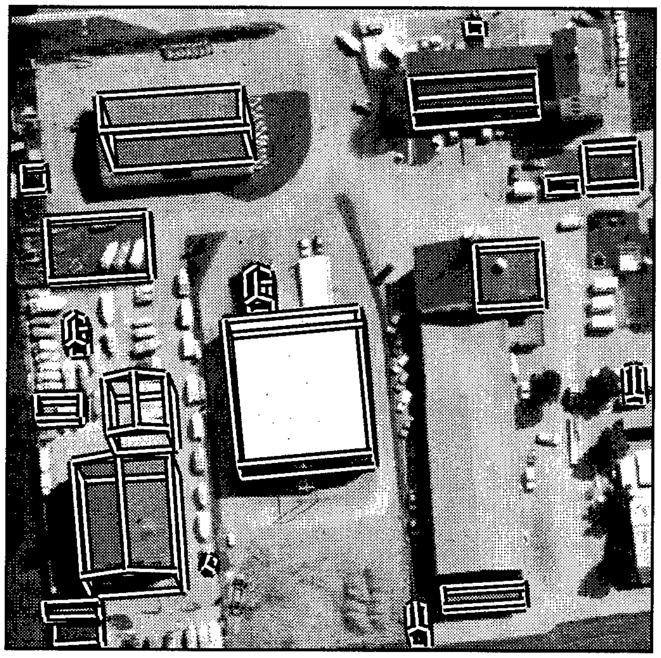


Figure B-4: PIVOT results, AVENCHINDUST1_4

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0.00	**0.000**	0.00	-	-	-
BUILD+SHAVE	0.00	**0.000**	0.00	0.00	**0.000**	0.00
VHBUILD	16.86	0.179	16.37	0.00	977.000	0.00
PIVOT	**56.66**	0.472	**44.71**	**46.47**	0.825	**33.59**

Table B-5: 2D and 3D evaluation results for AVENCHINDUST1_4

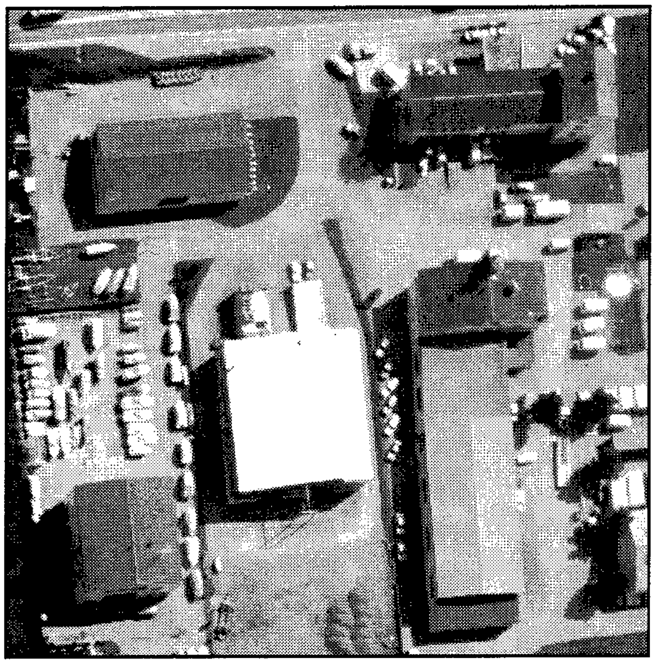


Figure B-5: AVENCHINDUST2_4 image

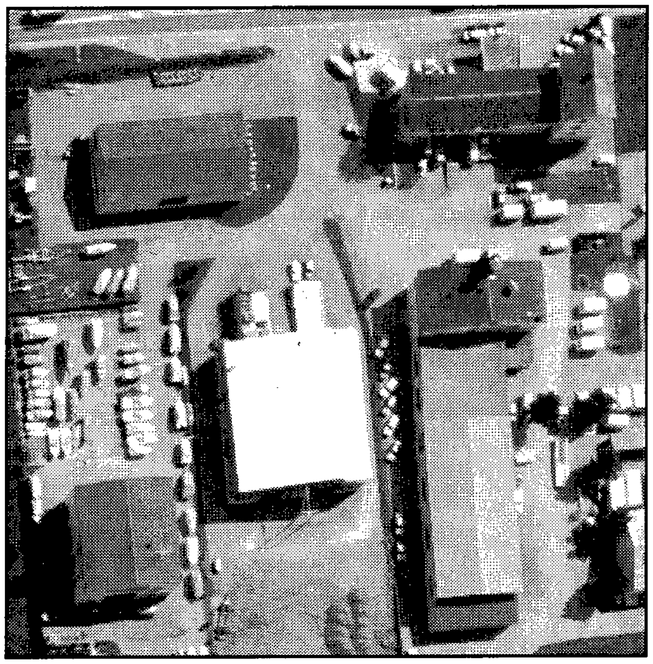


Figure B-6: BUILD+SHAVE results, AVENCHINDUST2_4

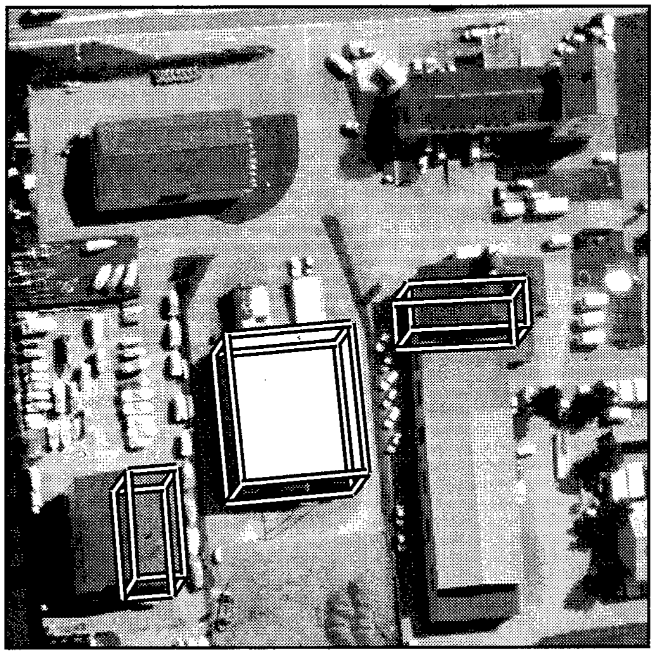


Figure B-7: VHBUILD results, AVENCHINDUST2_4

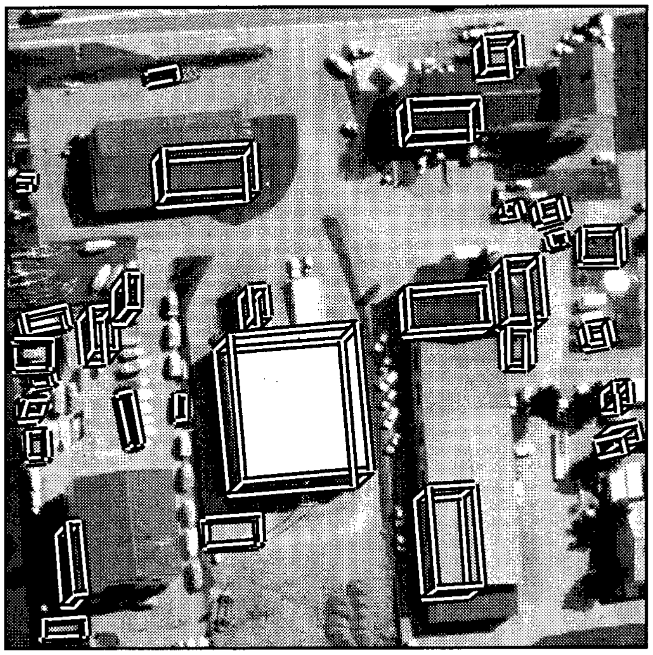


Figure B-8: PIVOT results, AVENCHINDUST2_4

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0.00	**0.000**	0.00	-	-	-
BUILD+SHAVE	0.00	**0.000**	0.00	0.00	**0.000**	0.00
VHBUILD	35.66	0.069	34.81	0.00	1408	0.00
PIVOT	**45.80**	0.485	**37.47**	**34.66**	0.370	**30.72**

Table B-6: 2D and 3D evaluation results for AVENCHINDUST2_4

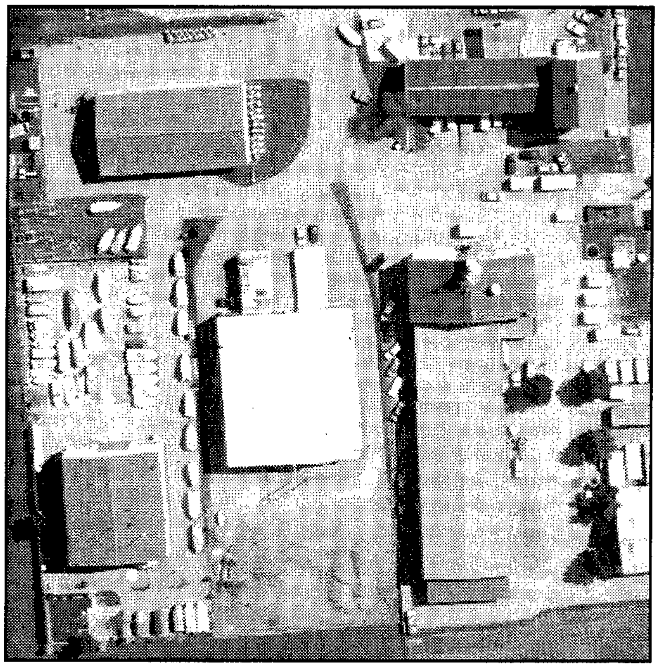


Figure B-9: AVENCHINDUST3_4 image

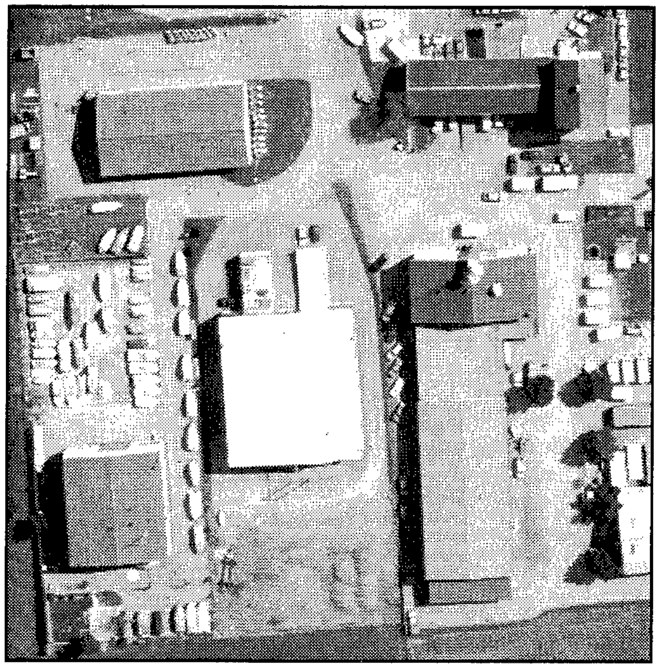


Figure B-10: BUILD+SHAVE results, AVENCHINDUST3_4

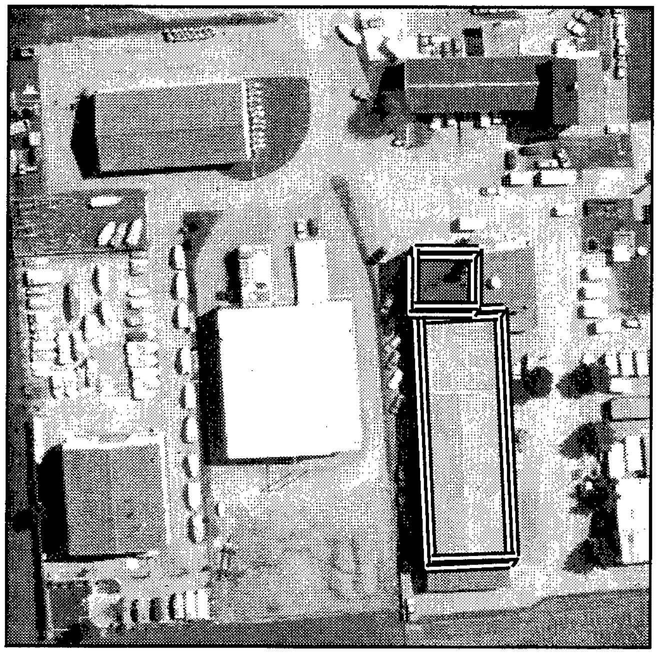


Figure B-11: VHBUILD results, AVENCHINDUST3_4

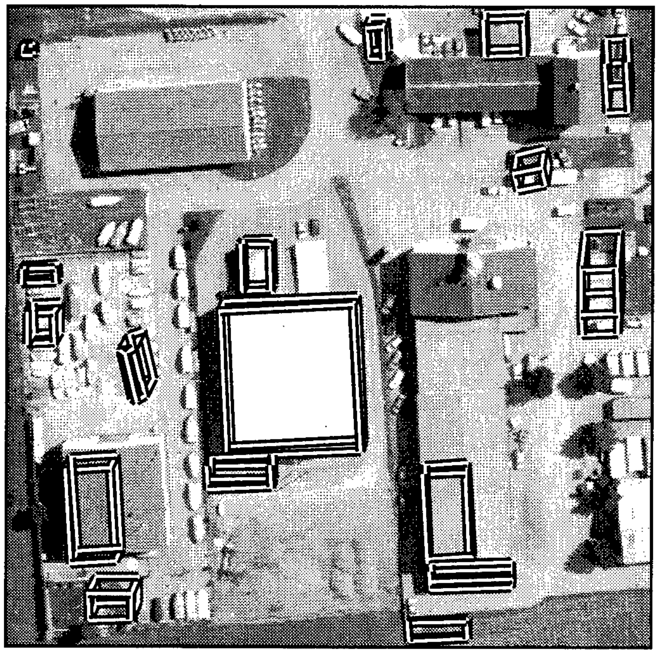


Figure B-12: PIVOT results, AVENCHINDUST3_4

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0.00	**0.000**	0.00	-	-	-
BUILD+SHAVE	0.00	**0.000**	0.00	0.00	**0.000**	0.00
VHBUILD	28.68	0.033	28.41	0.40	0.292	0.40
PIVOT	**35.53**	0.640	**28.95**	**30.57**	1.038	**23.21**

Table B-7: 2D and 3D evaluation results for AVENCHINDUST3_4

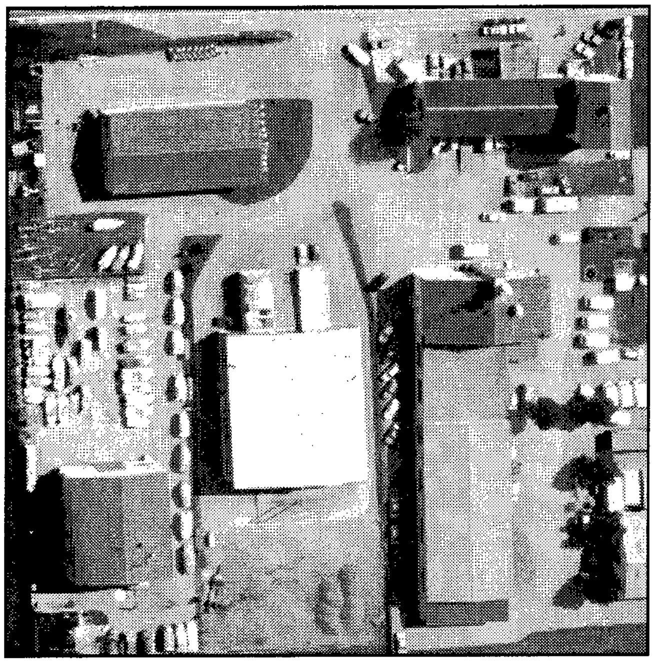


Figure B-13: AVENCHINDUST4_4 image

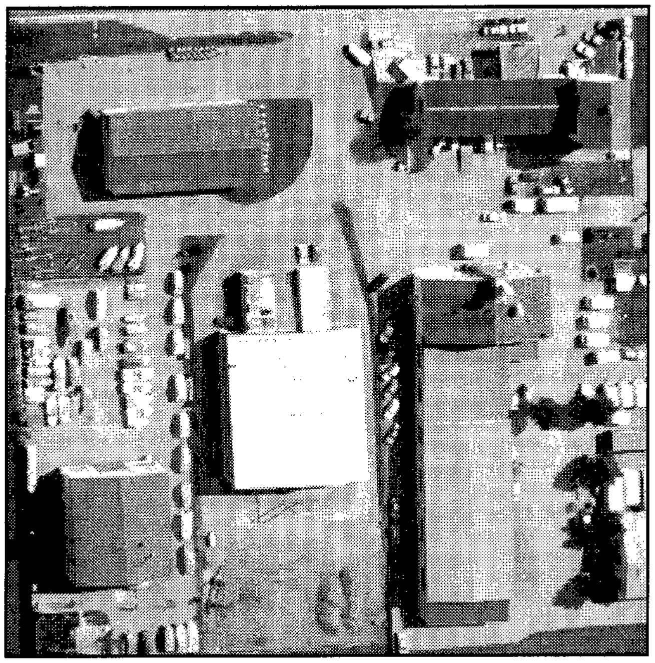


Figure B-14: BUILD+SHAVE results, AVENCHINDUST4_4

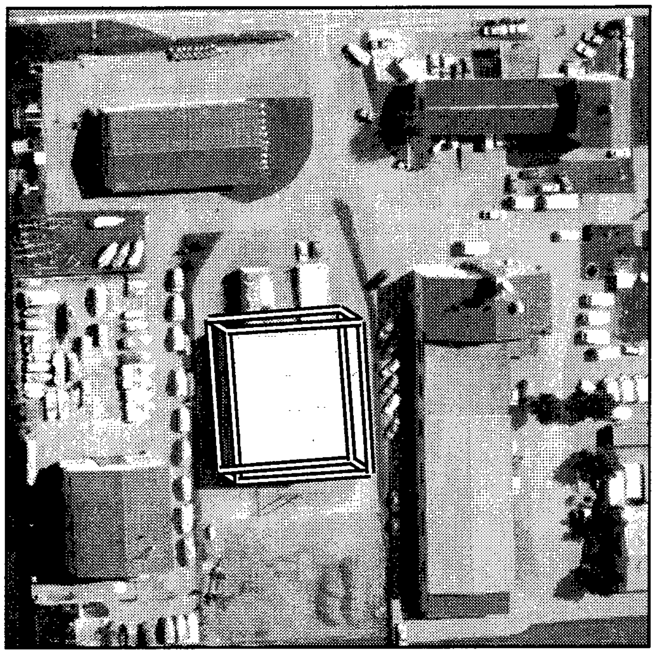


Figure B-15: VHBUILD results, AVENCHINDUST4_4

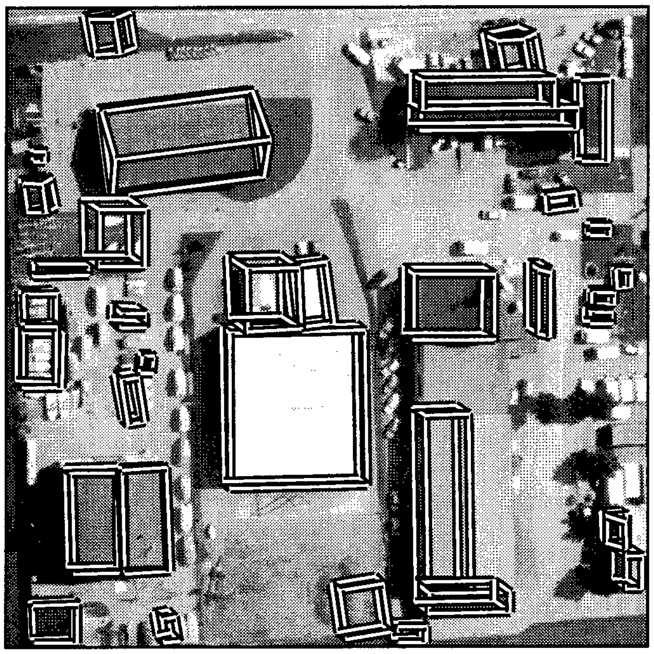


Figure B-16: PIVOT results, AVENCHINDUST4_4

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0.00	**0.000**	0.00	-	-	-
BUILD+SHAVE	0.00	**0.000**	0.00	0.00	**0.000**	0.00
VHBUILD	22.54	0.098	22.05	0.00	1380	0.00
PIVOT	**76.71**	0.522	**54.79**	**47.55**	0.641	**36.44**

Table B-8: 2D and 3D evaluation results for AVENCHINDUST4_4

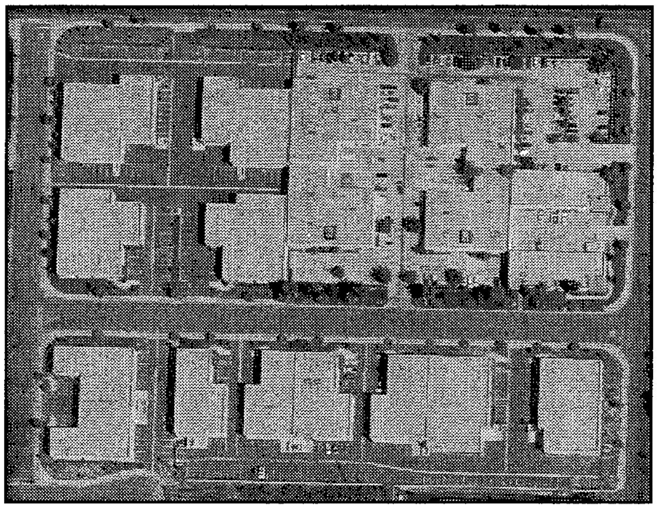


Figure B-17: BLOCK_RI0707 image

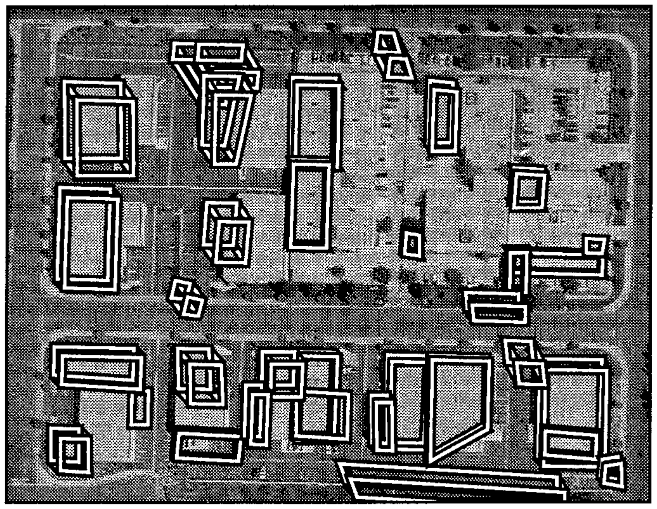


Figure B-18: BUILD+SHAVE results, BLOCK_RI0707

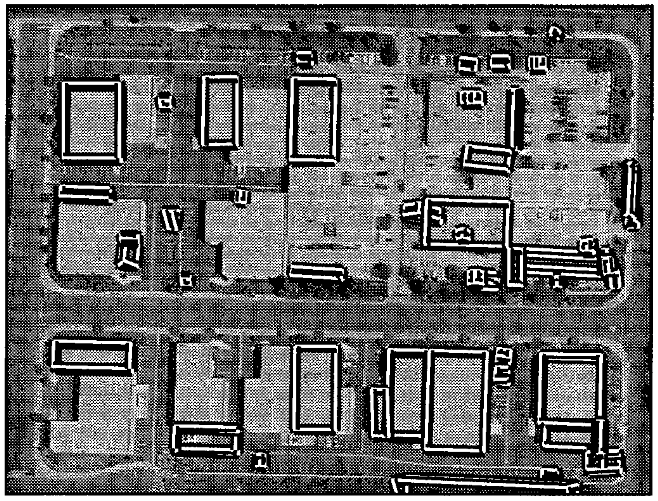


Figure B-19: VHBUILD results, BLOCK_RI0707

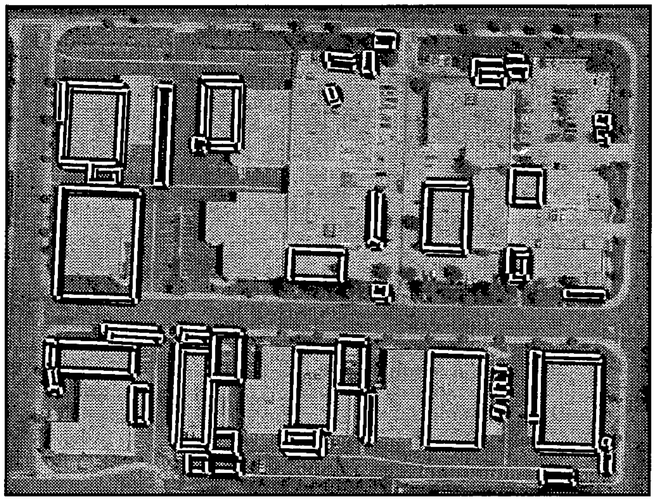


Figure B-20: PIVOT results, BLOCK_RI0707

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	49.76	**0.202**	45.23	-	-	-
BUILD+SHAVE	**58.31**	0.381	**47.71**	**45.16**	2.124	**23.05**
VHBUILD	42.54	0.256	38.36	19.99	1.806	14.69
PIVOT	41.06	0.528	33.75	25.12	**1.039**	19.92

Table B-9: 2D and 3D evaluation results for BLOCK_RI0707

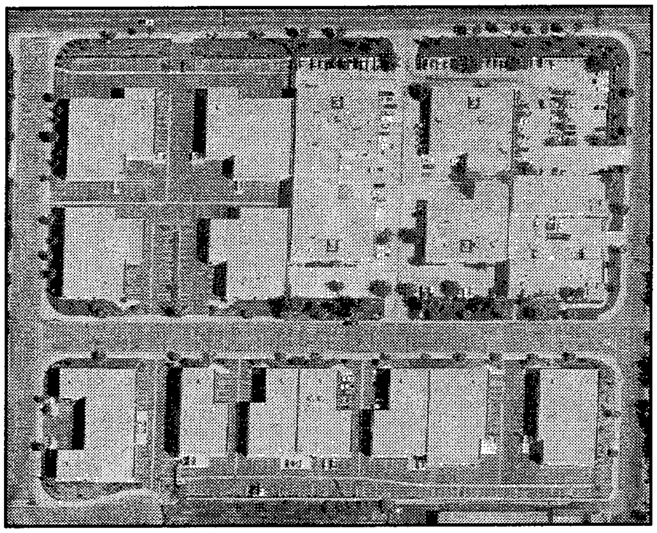


Figure B-21: BLOCK_RI0808 image

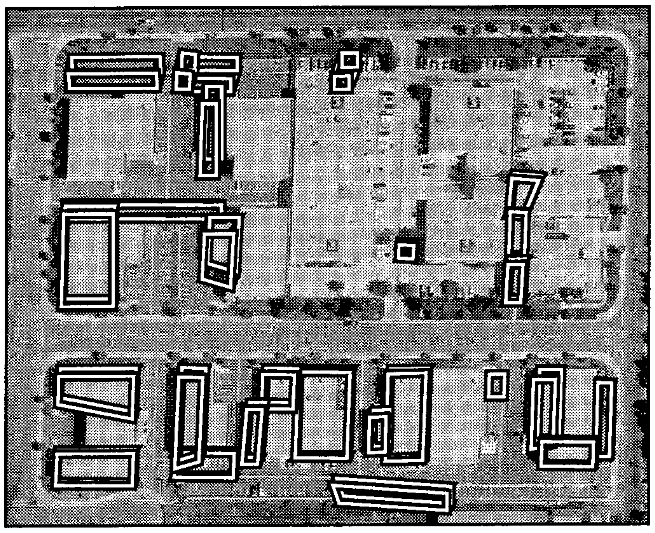


Figure B-22: BUILD+SHAVE results, BLOCK_RI0808

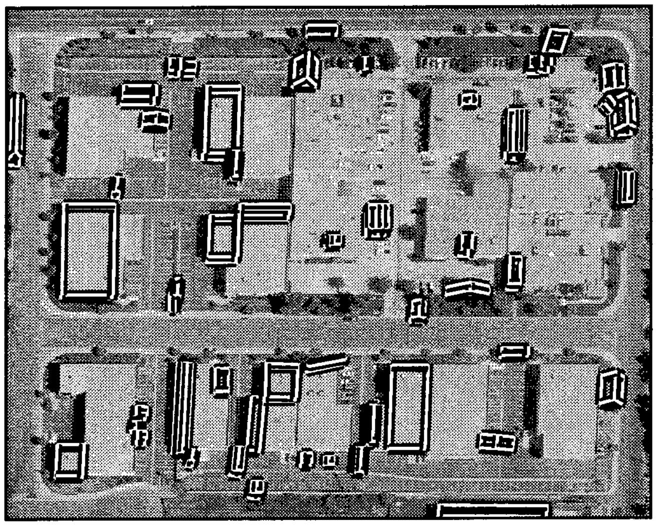


Figure B-23: VHBUILD results, BLOCK_RI0808

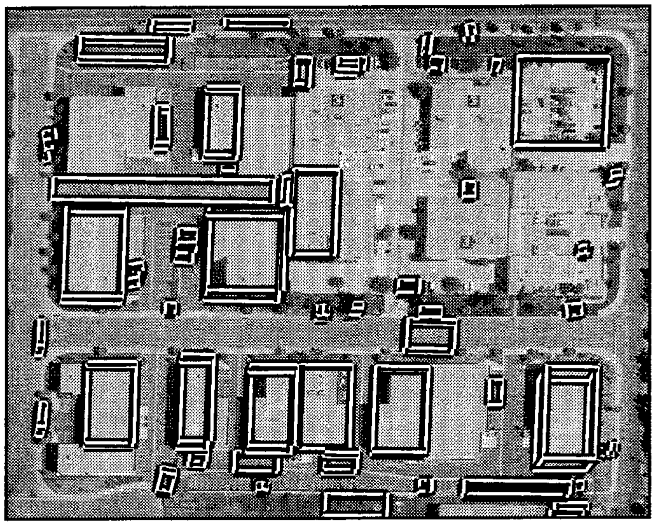


Figure B-24: PIVOT results, BLOCK_RI0808

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	30.04	**0.328**	27.35	-	-	-
BUILD+SHAVE	32.40	0.527	27.67	26.37	1.403	19.25
VHBUILD	20.89	0.660	18.36	13.59	**0.975**	12.00
PIVOT	**44.31**	0.846	**32.23**	**33.50**	1.614	**21.74**

Table B-10: 2D and 3D evaluation results for BLOCK_RI0808

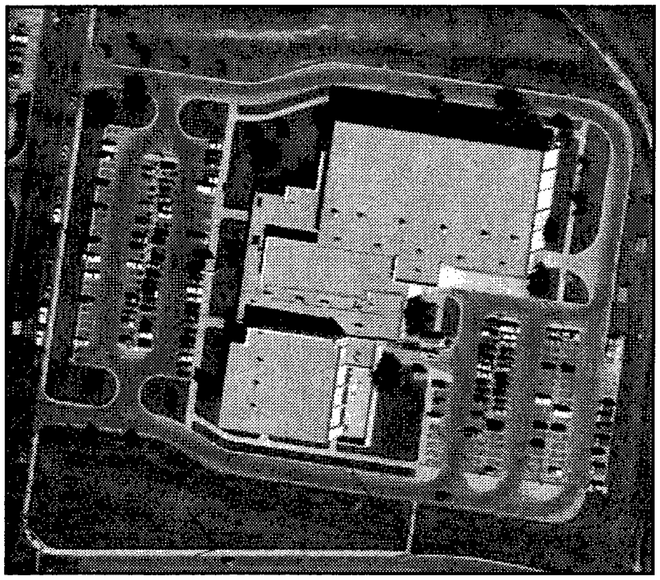


Figure B-25: COMPLEX_FHN713 image

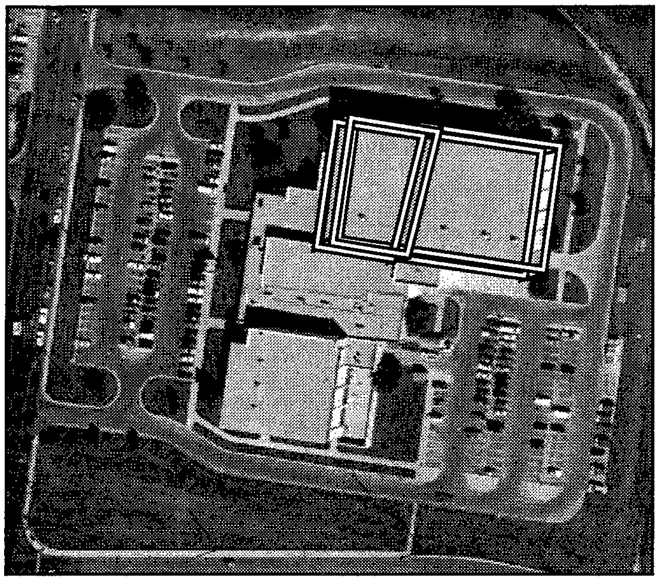


Figure B-26: BUILD+SHAVE results, COMPLEX_FHN713

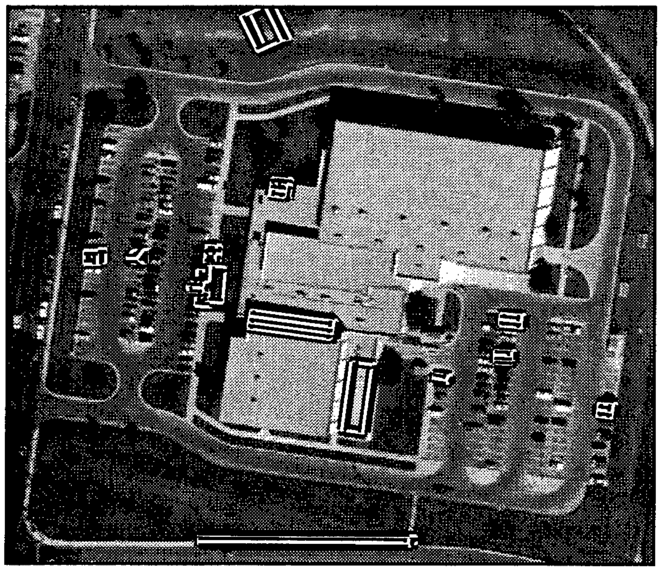


Figure B-27: VHBUILD results, COMPLEX_FHN713

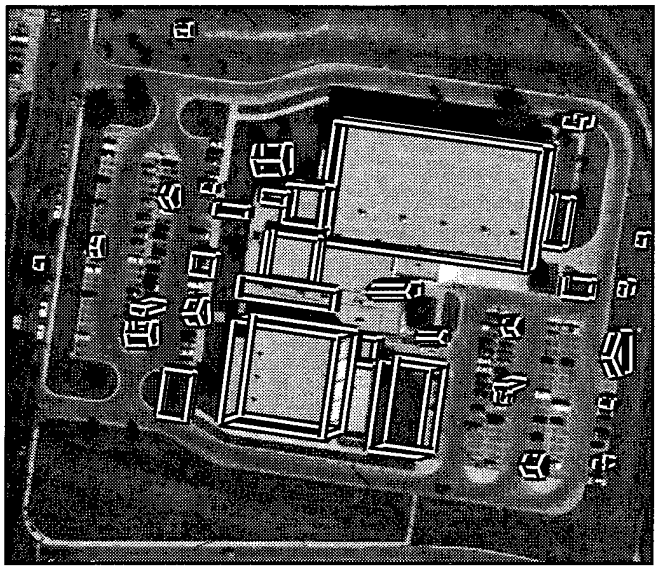


Figure B-28: PIVOT results, COMPLEX_FHN713

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.38	**0.000**	42.38	-	-	-
BUILD+SHAVE	47.94	0.014	47.61	46.22	**0.248**	41.46
VHBUILD	3.27	4.280	2.87	0.66	6.960	0.63
PIVOT	**79.07**	0.413	**59.61**	**49.68**	0.350	**42.32**

Table B-11: 2D and 3D evaluation results for COMPLEX_FHN713

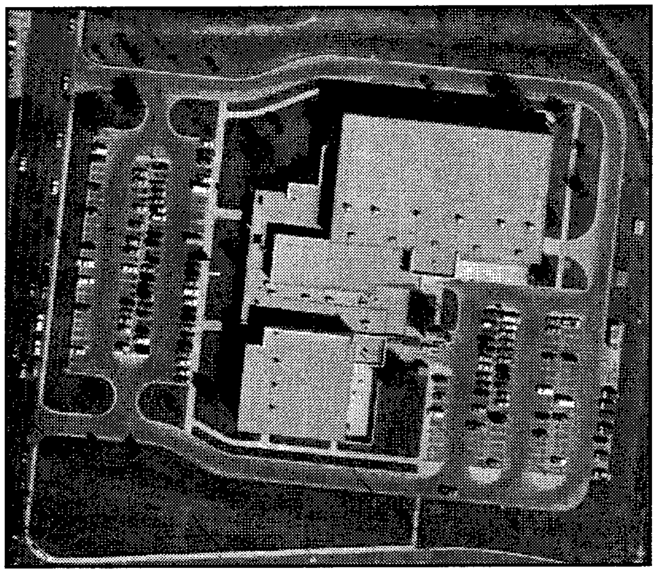


Figure B-29: COMPLEX_FHN715 image

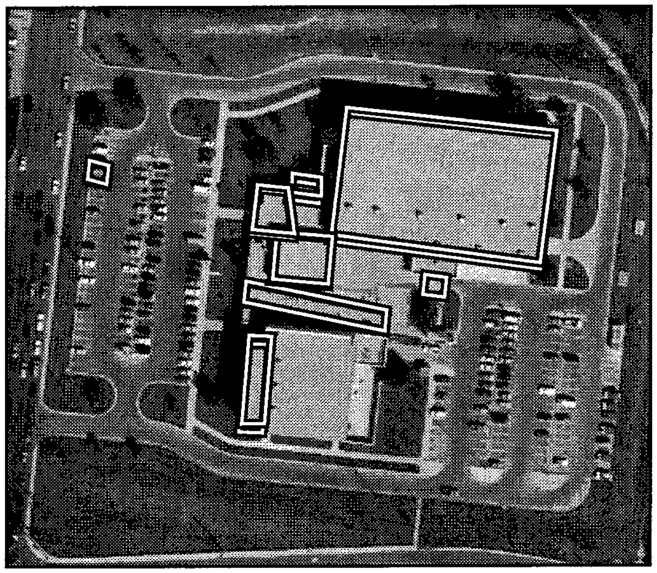


Figure B-30: BUILD+SHAVE results, COMPLEX_FHN715

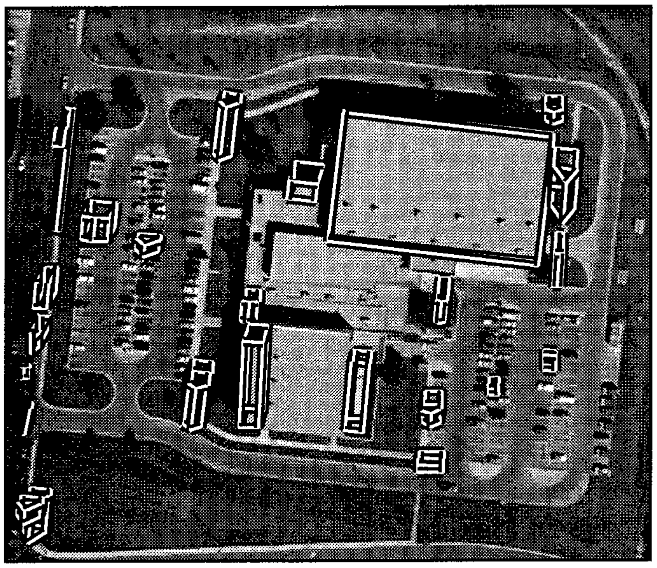


Figure B-31: VHBUILD results, COMPLEX_FHN715

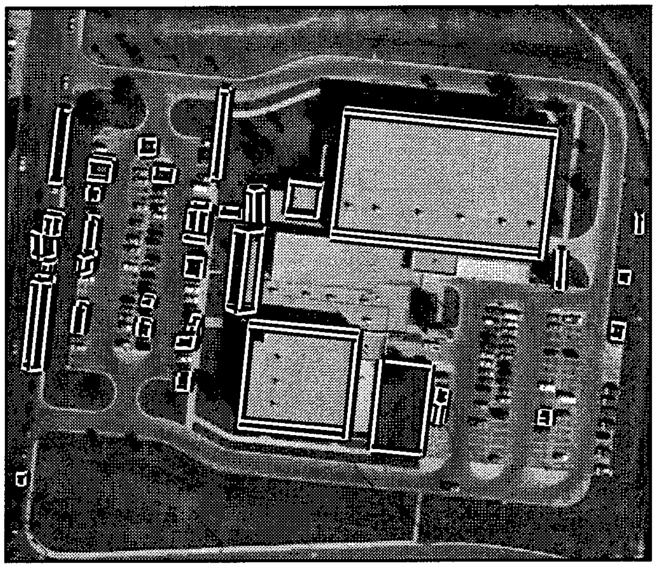


Figure B-32: PIVOT results, COMPLEX_FHN715

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	63.42	**0.040**	**61.84**	-	-	-
BUILD+SHAVE	64.65	0.076	61.62	43.69	0.668	33.82
VHBUILD	53.01	0.547	41.10	36.26	1.519	23.38
PIVOT	**68.64**	0.521	50.57	**66.97**	**0.554**	**48.85**

Table B-12: 2D and 3D evaluation results for COMPLEX_FHN715

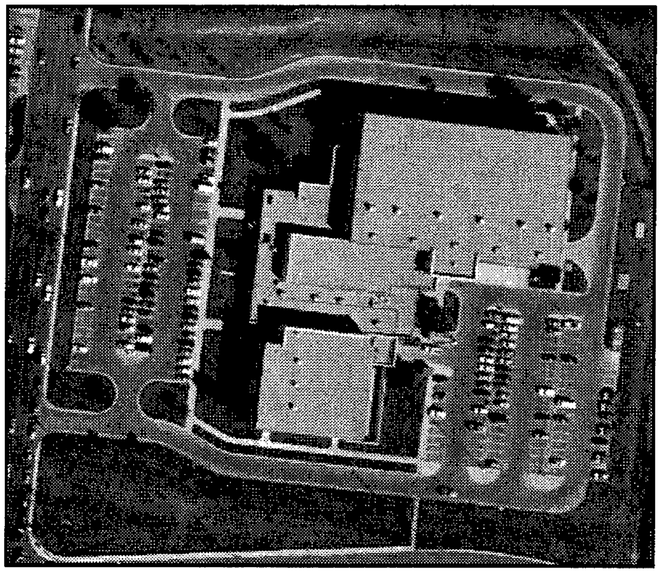


Figure B-33: COMPLEX_FHN717 image

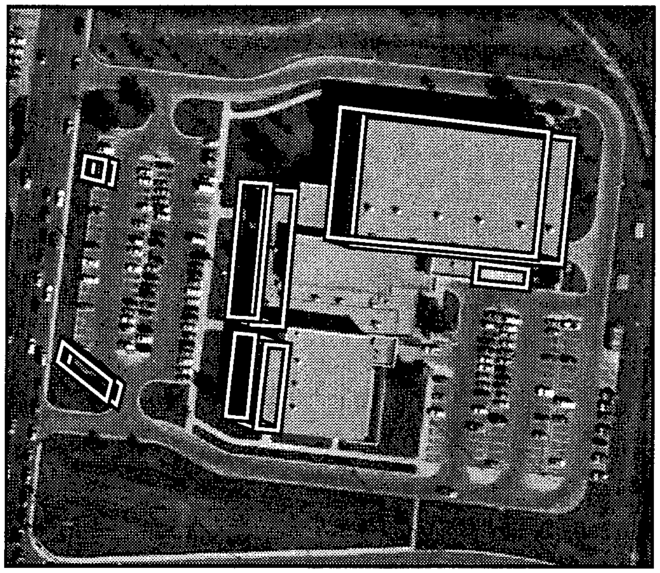


Figure B-34: BUILD+SHAVE results, COMPLEX_FHN717

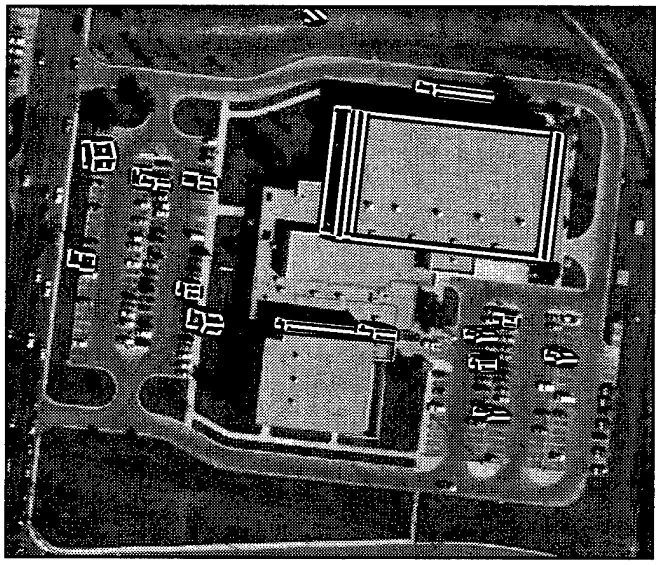


Figure B-35: VHBUILD results, COMPLEX_FHN717

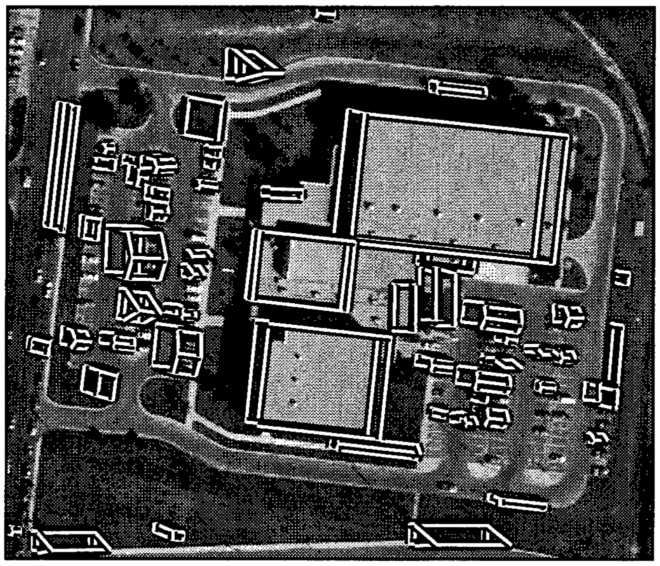


Figure B-36: PIVOT results, COMPLEX_FHN717

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	51.68	**0.091**	49.35	-	-	-
BUILD+SHAVE	60.61	0.209	**53.81**	39.00	1.525	24.46
VHBUILD	46.91	0.226	42.41	22.68	**0.353**	21.00
PIVOT	**81.66**	0.723	51.35	**65.39**	0.799	**42.95**

Table B-13: 2D and 3D evaluation results for COMPLEX_FHN717

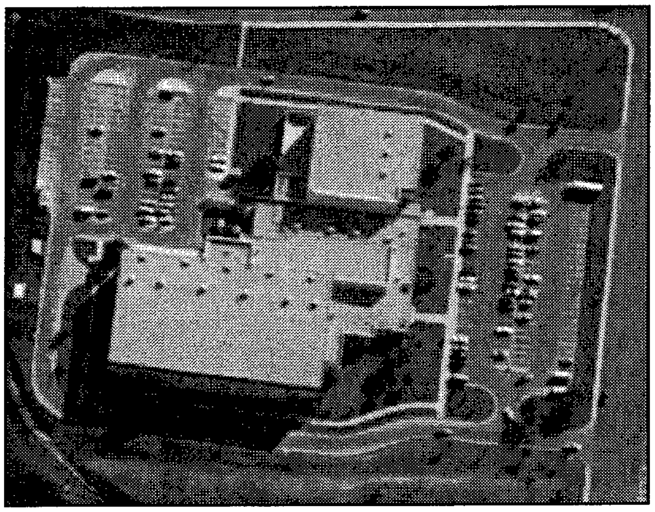


Figure B-37: COMPLEX_FHOV1627 image

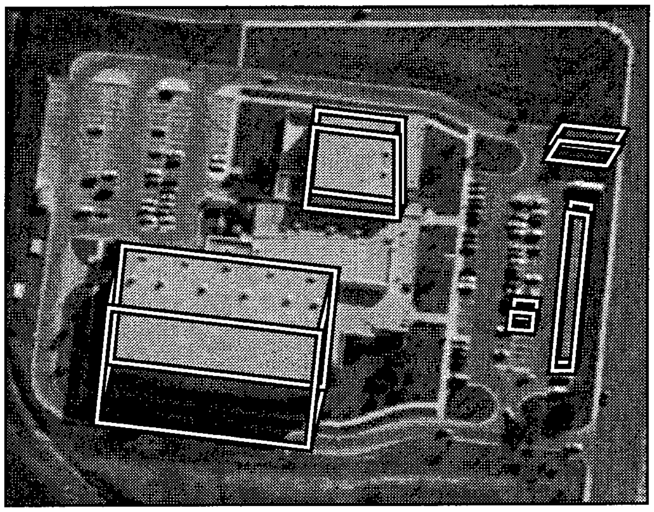


Figure B-38: BUILD+SHAVE results, COMPLEX_FHOV1627

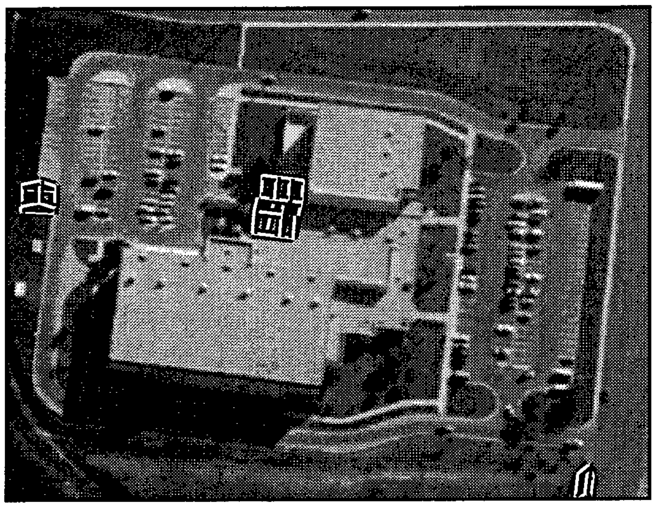


Figure B-39: VHBUILD results, COMPLEX_FHOV1627

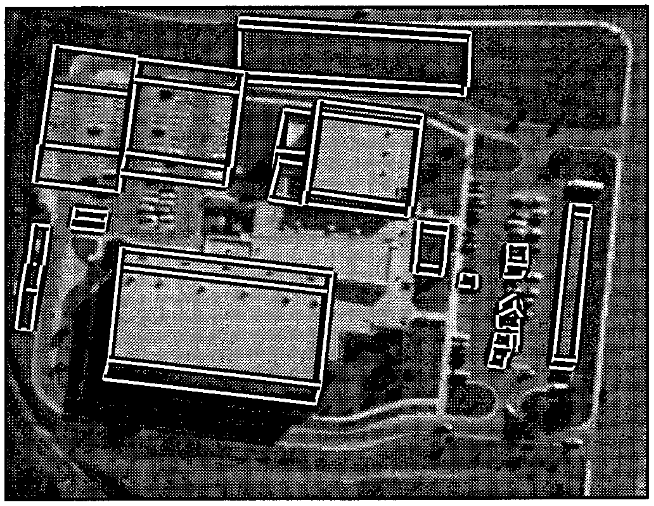


Figure B-40: PIVOT results, COMPLEX_FHOV1627

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	56.94	**0.172**	**51.86**	-	-	-
BUILD+SHAVE	67.62	0.450	51.84	42.04	3.462	17.12
VHBUILD	2.65	1.837	2.53	0.87	7.039	0.82
PIVOT	**68.73**	1.412	34.89	**56.19**	**1.863**	**27.45**

Table B-14: 2D and 3D evaluation results for COMPLEX_FHOV1627

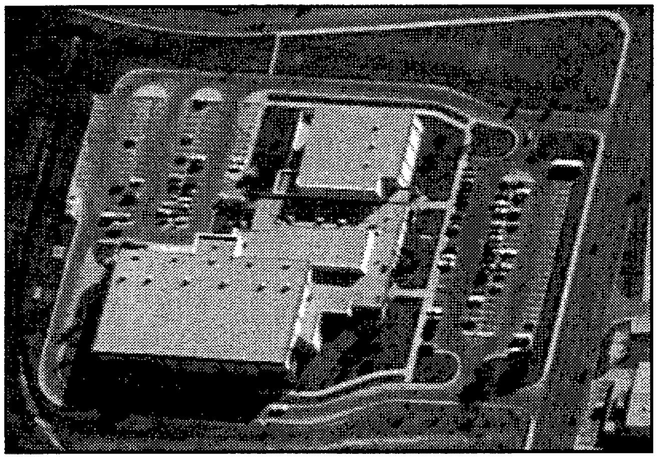


Figure B-41: COMPLEX_FHOV1727 image

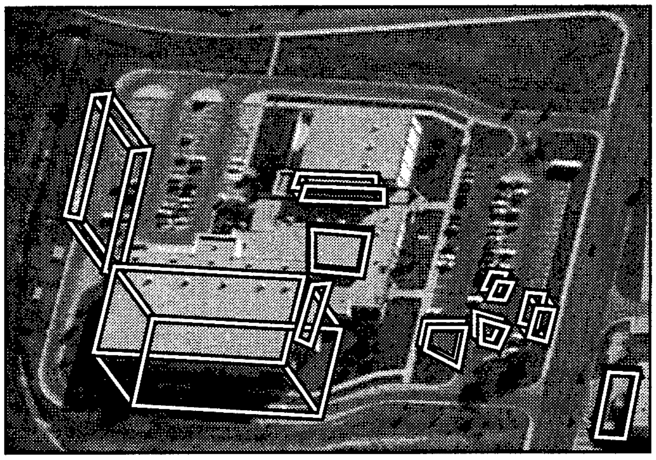


Figure B-42: BUILD+SHAVE results, COMPLEX_FHOV1727

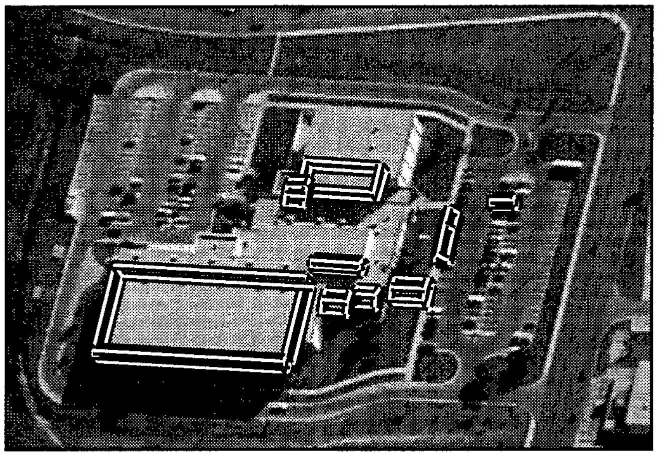


Figure B-43: VHBUILD results, COMPLEX_FHOV1727

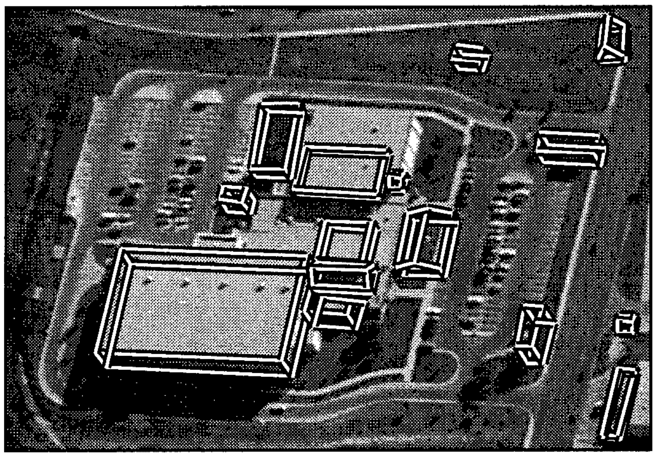


Figure B-44: PIVOT results, COMPLEX_FHOV1727

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.83	0.256	38.60	-	-	-
BUILD+SHAVE	55.08	0.790	38.38	24.55	5.464	10.49
VHBUILD	47.41	**0.108**	45.10	31.98	**0.071**	31.28
PIVOT	**67.54**	0.385	**53.59**	**49.70**	0.335	**42.60**

Table B-15: 2D and 3D evaluation results for COMPLEX_FHOV1727

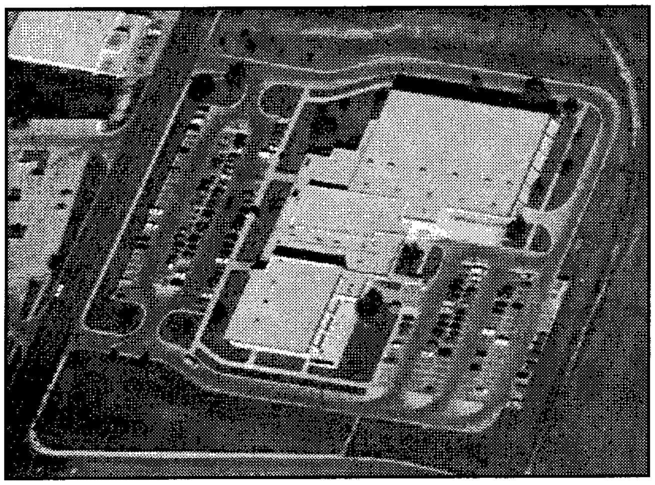


Figure B-45: COMPLEX_FHOV525 image

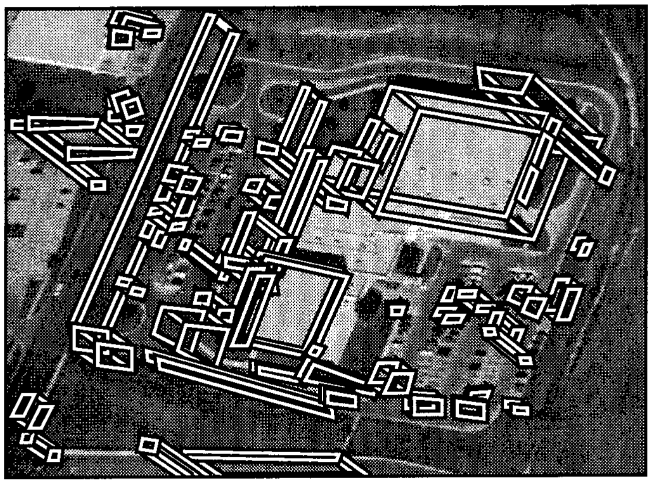


Figure B-46: BUILD+SHAVE results, COMPLEX_FHOV525

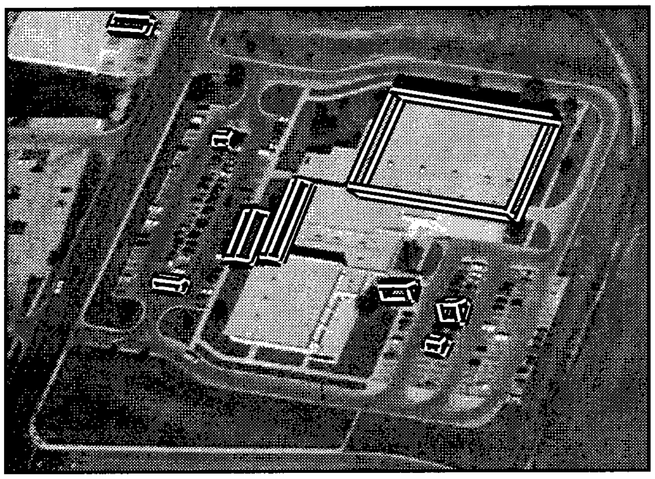


Figure B-47: VHBUILD results, COMPLEX_FHOV525

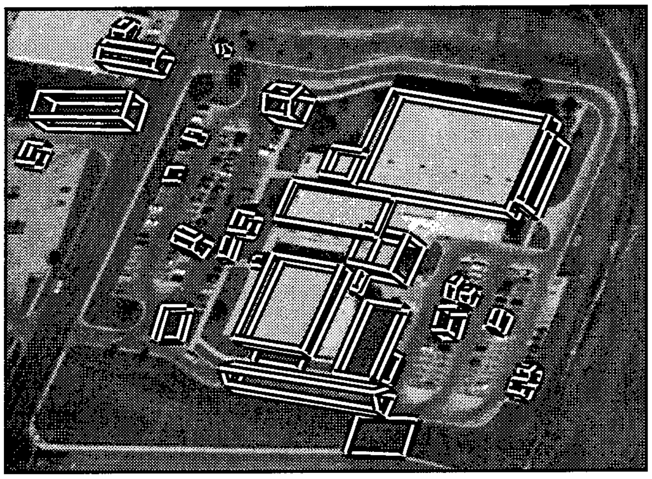


Figure B-48: PIVOT results, COMPLEX_FHOV525

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	55.03	0.691	39.87	-	-	-
BUILD+SHAVE	70.19	1.635	32.68	46.89	4.129	15.97
VHBUILD	43.06	**0.191**	39.79	36.16	**0.289**	32.74
PIVOT	**71.41**	0.777	**45.94**	**52.37**	0.985	**34.55**

Table B-16: 2D and 3D evaluation results for COMPLEX_FHOV525

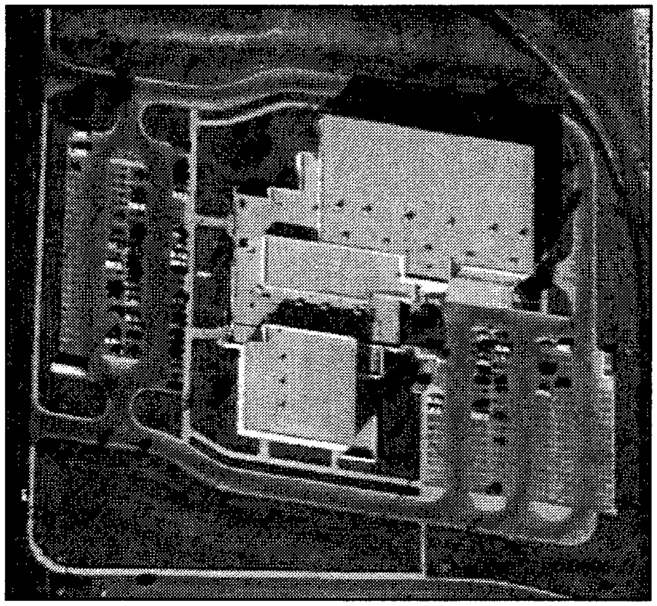


Figure B-49: COMPLEX_FHOV727 image

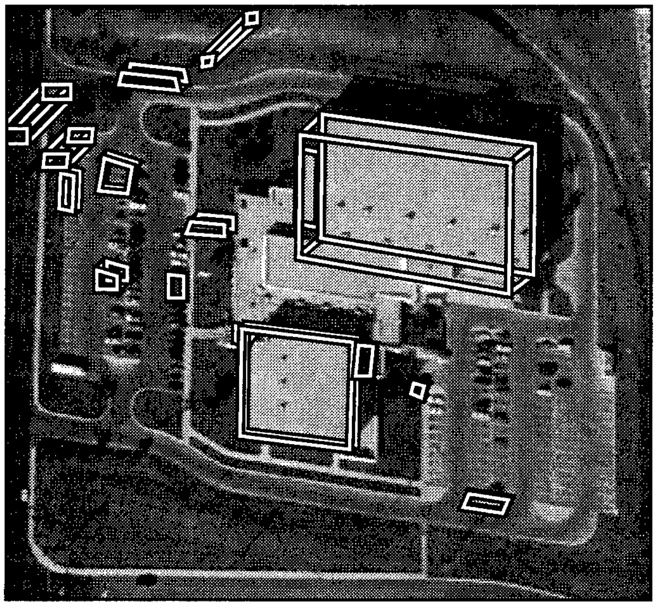


Figure B-50: BUILD+SHAVE results, COMPLEX_FHOV727

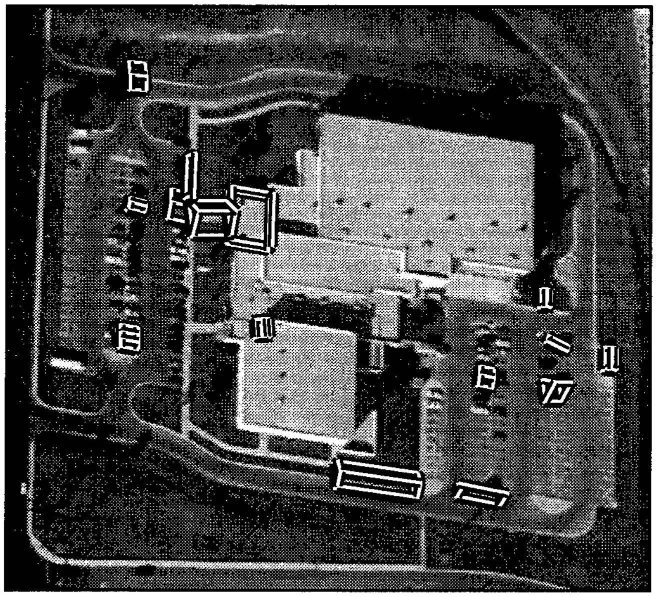


Figure B-51: VHBUILD results, COMPLEX_FHOV727

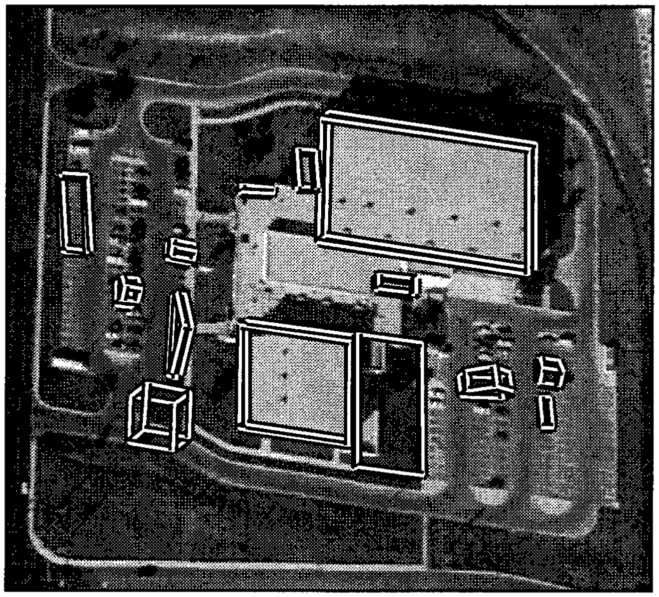


Figure B-52: PIVOT results, COMPLEX_FHOV727

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	63.38	**0.155**	57.72	-	-	-
BUILD+SHAVE	**72.87**	0.277	**60.63**	**52.58**	1.845	26.69
VHBUILD	4.32	3.464	3.76	1.43	5.609	1.33
PIVOT	67.75	0.500	50.61	51.85	**0.360**	**43.70**

Table B-17: 2D and 3D evaluation results for COMPLEX_FHOV727

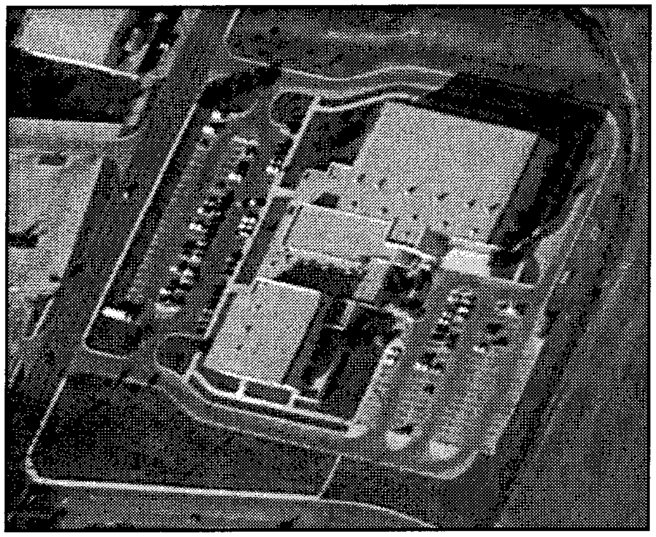


Figure B-53: COMPLEX_FHOV927 image

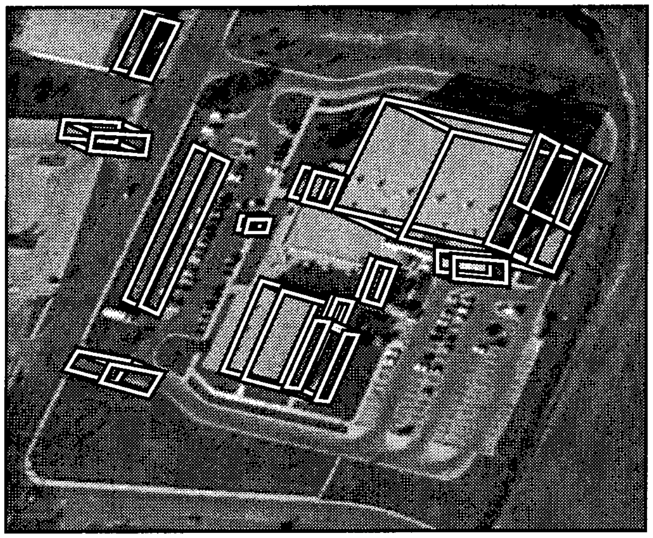


Figure B-54: BUILD+SHAVE results, COMPLEX_FHOV927

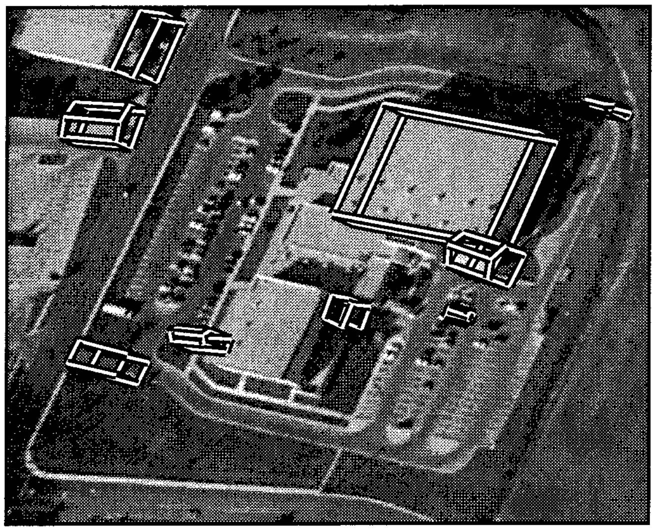


Figure B-55: VHBUILD results, COMPLEX_FHOV927

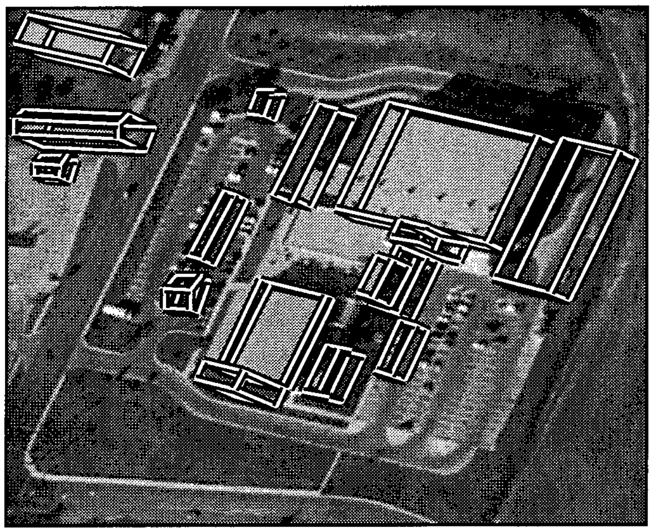


Figure B-56: PIVOT results, COMPLEX_FHOV927

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	58.11	**0.251**	**50.71**	-	-	-
BUILD+SHAVE	66.63	0.640	46.71	39.07	3.611	16.21
VHBUILD	46.18	0.433	38.49	36.52	**0.597**	29.98
PIVOT	**70.12**	0.897	43.04	**50.45**	1.170	**31.73**

Table B-18: 2D and 3D evaluation results for COMPLEX_FHOV927

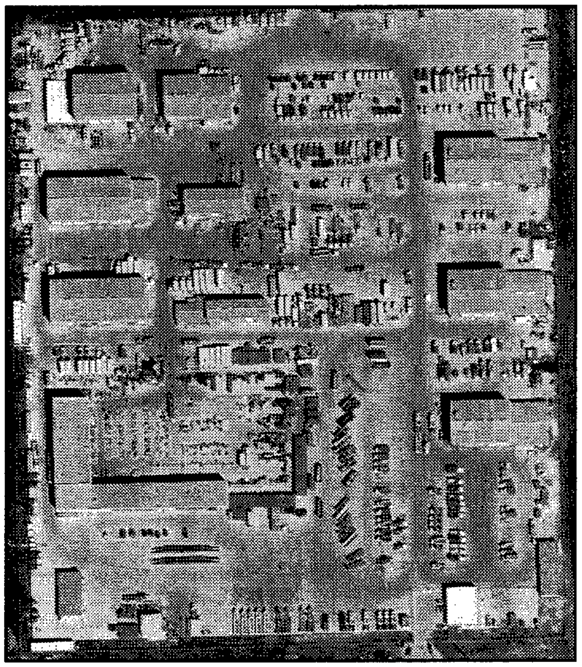


Figure B-57: DARK_FHN717 image

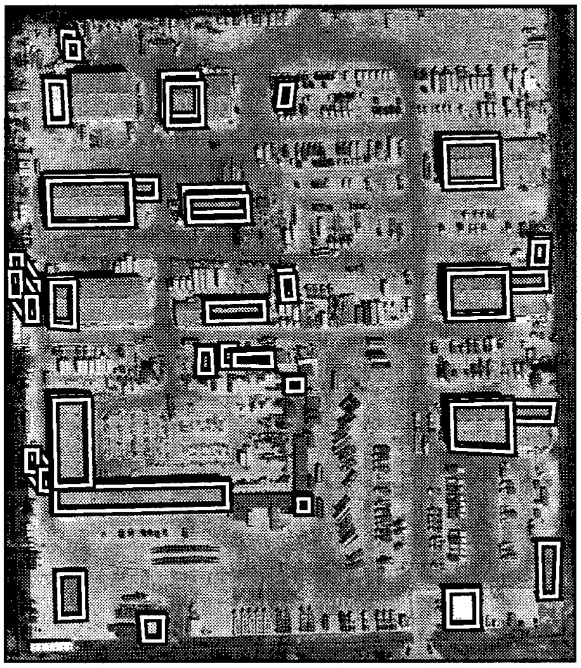


Figure B-58: BUILD+SHAVE results, DARK_FHN717

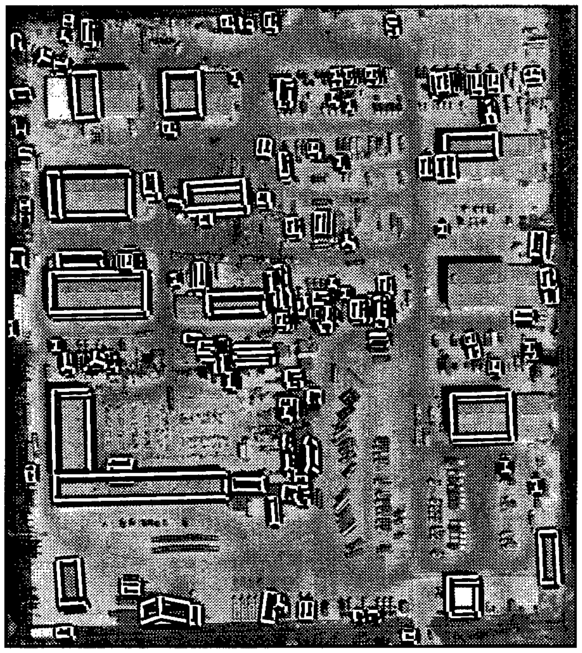


Figure B-59: VHBUILD results, DARK_FHN717

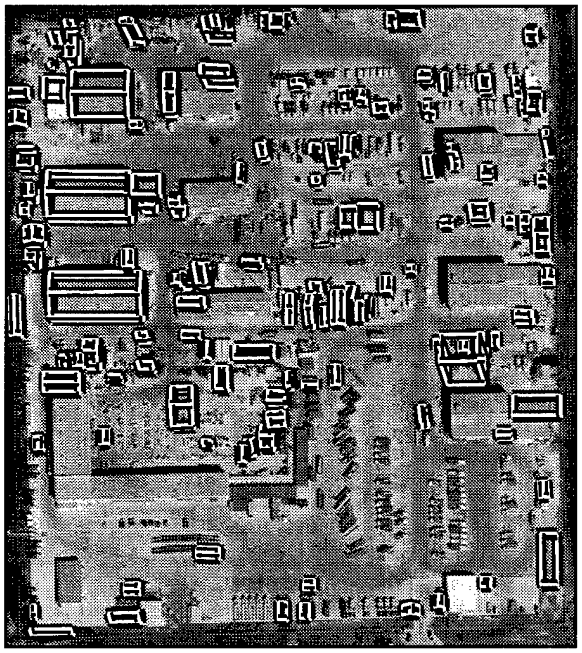


Figure B-60: PIVOT results, DARK_FHN717

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	62.38	**0.118**	58.10	-	-	-
BUILD+SHAVE	**68.61**	0.206	**60.10**	**37.60**	**0.732**	**29.48**
VHBUILD	63.20	0.797	42.03	36.99	1.173	25.80
PIVOT	34.24	1.715	21.57	26.18	2.019	17.13

Table B-19: 2D and 3D evaluation results for DARK_FHN717

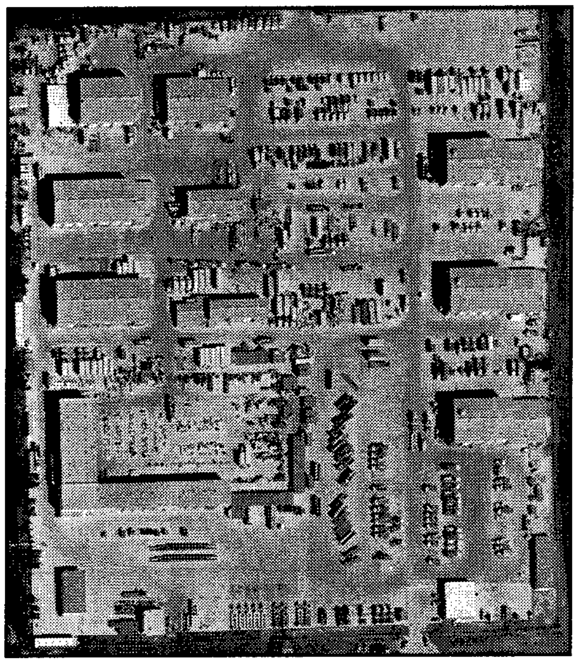


Figure B-61: DARK_FHN719 image

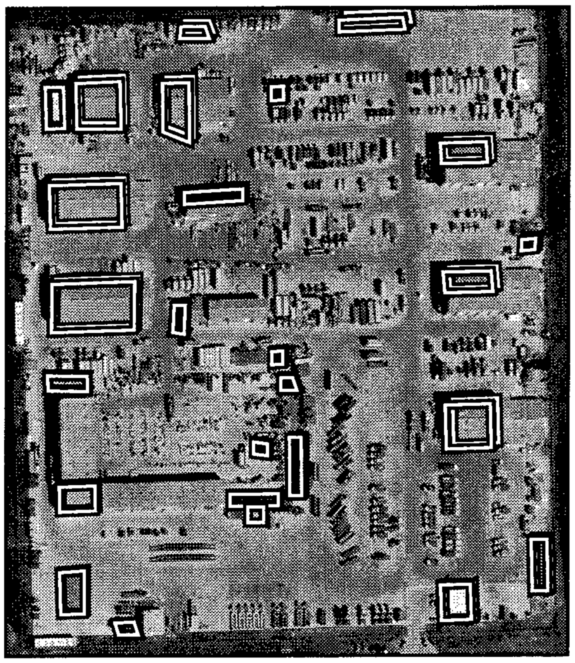


Figure B-62: BUILD+SHAVE results, DARK_FHN719

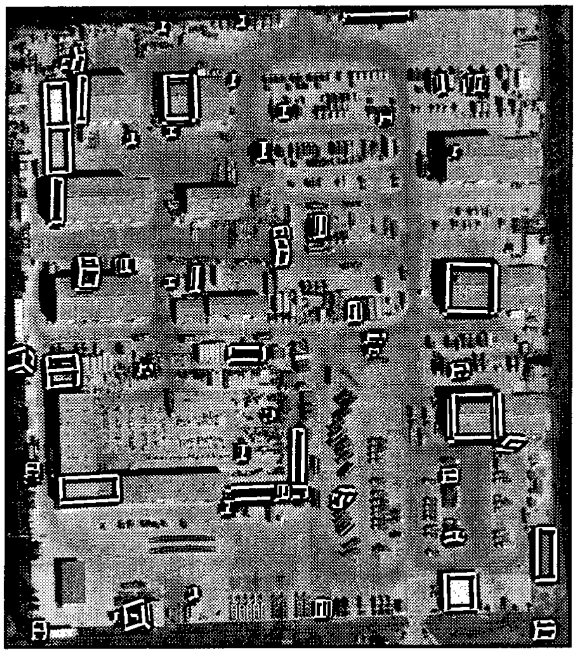


Figure B-63: VHBUILD results, DARK_FHN719

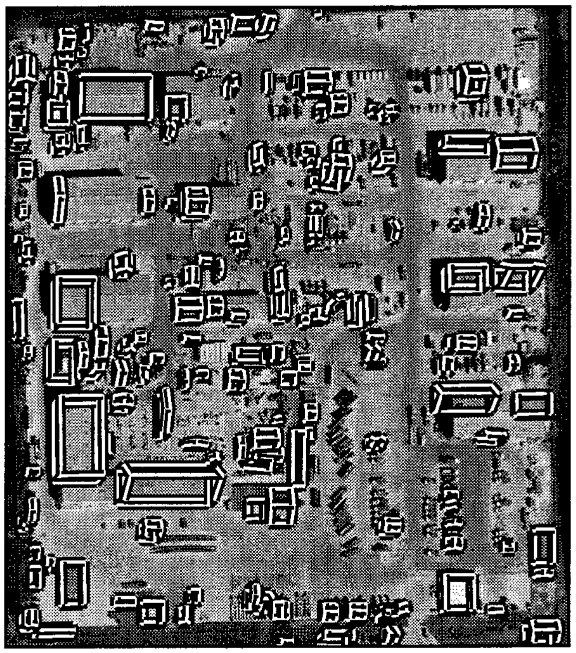


Figure B-64: PIVOT results, DARK_FHN719

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.06	**0.239**	38.22	-	-	-
BUILD+SHAVE	48.73	0.318	**42.20**	33.57	**0.356**	**29.98**
VHBUILD	26.52	0.995	20.98	11.13	1.167	9.85
PIVOT	**53.28**	1.539	29.28	**34.18**	2.051	20.09

Table B-20: 2D and 3D evaluation results for DARK_FHN719

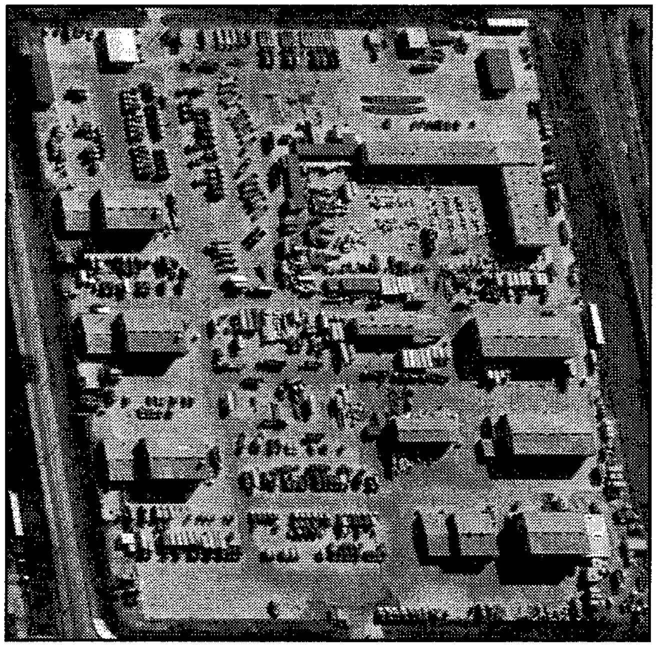


Figure B-65: DARK_FHOV1627 image

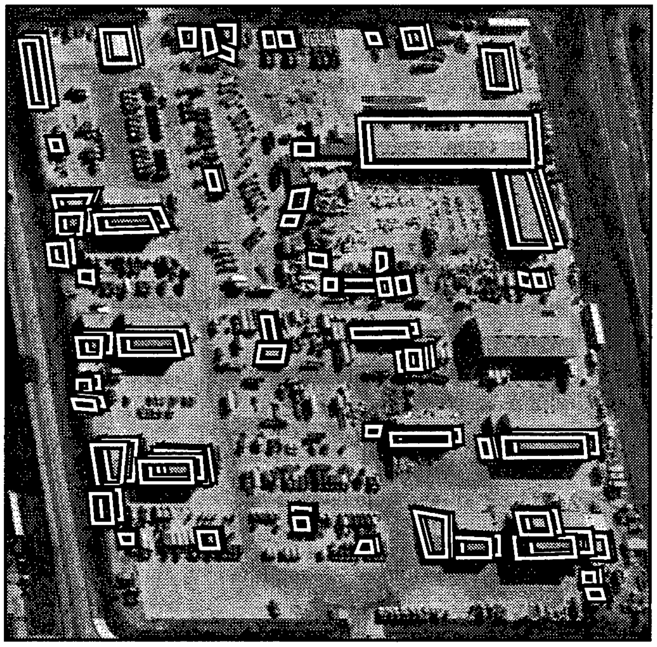


Figure B-66: BUILD+SHAVE results, DARK_FHOV1627

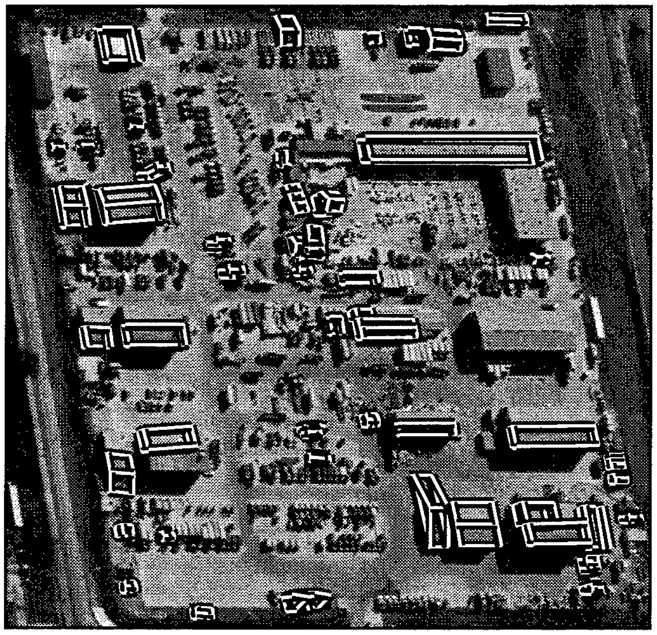


Figure B-67: VHBUILD results, DARK_FHOV1627

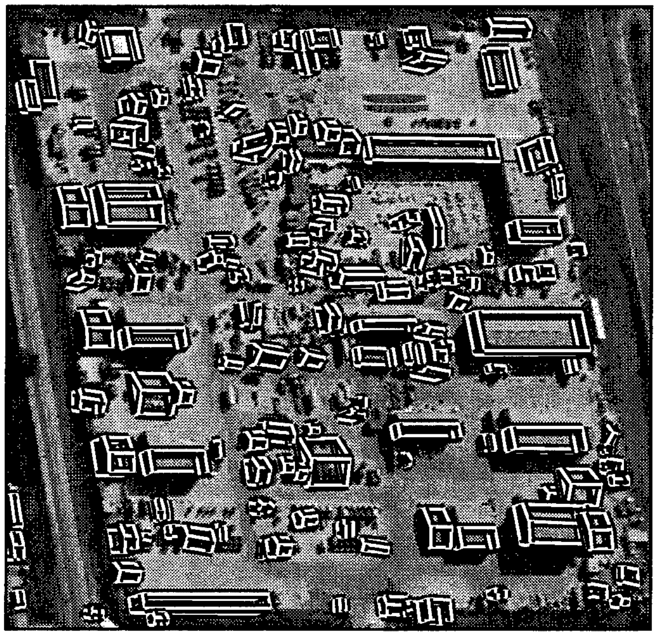


Figure B-68: PIVOT results, DARK_FHOV1627

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	50.53	0.573	39.19	-	-	-
BUILD+SHAVE	65.30	0.589	**47.16**	47.75	0.901	**33.38**
VHBUILD	49.50	**0.560**	38.76	29.37	**0.684**	24.45
PIVOT	**68.51**	1.532	33.42	**50.88**	1.656	27.61

Table B-21: 2D and 3D evaluation results for DARK_FHOV1627

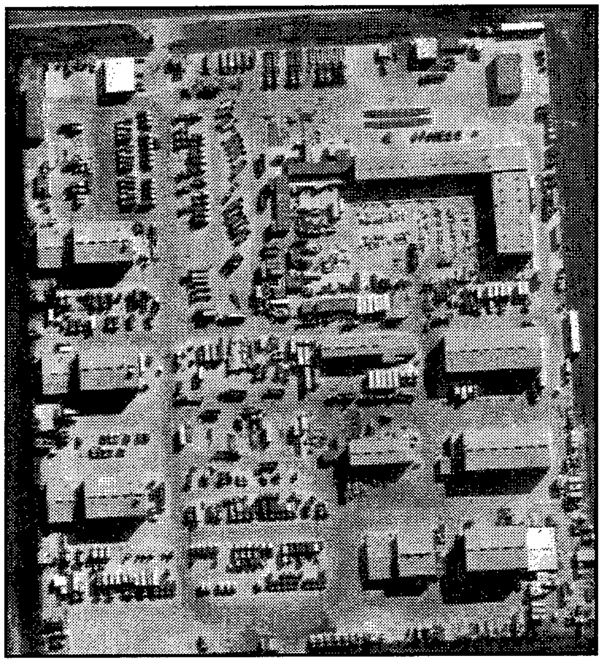


Figure B-69: DARK_FHOV1727 image

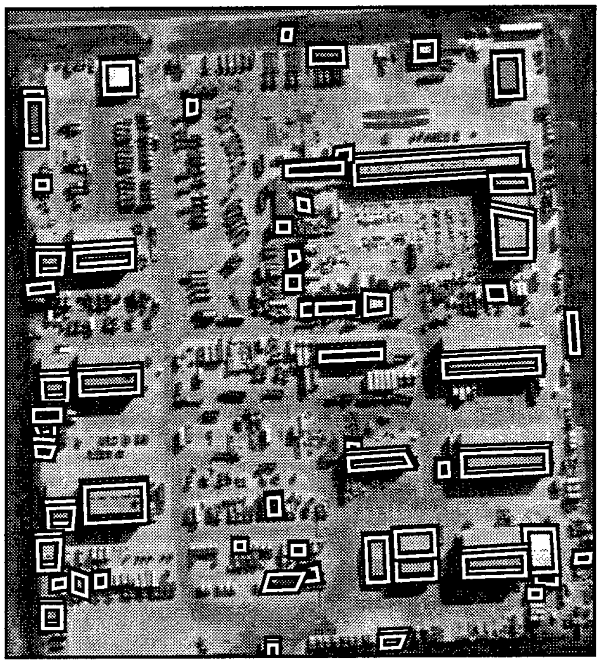


Figure B-70: BUILD+SHAVE results, DARK_FHOV1727

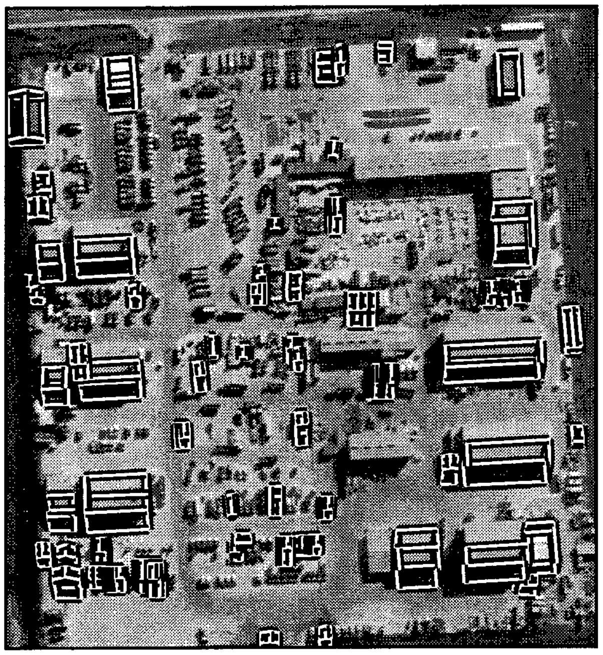


Figure B-71: VHBUILD results, DARK_FHOV1727

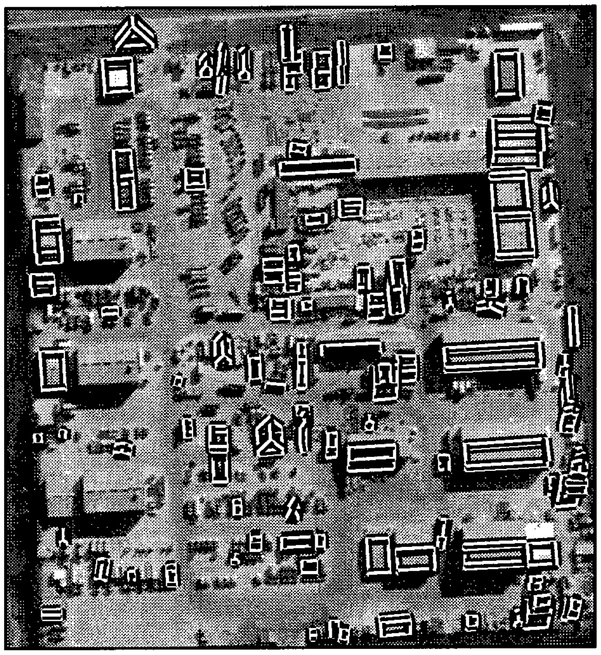


Figure B-72: PIVOT results, DARK_FHOV1727

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	57.85	**0.370**	47.66	-	-	-
BUILD+SHAVE	**67.22**	0.393	**53.18**	**53.52**	**0.622**	**40.15**
VHBUILD	49.81	1.200	31.18	22.79	10.273	6.82
PIVOT	40.45	1.566	24.76	29.77	4.222	13.19

Table B-22: 2D and 3D evaluation results for DARK_FHOV1727

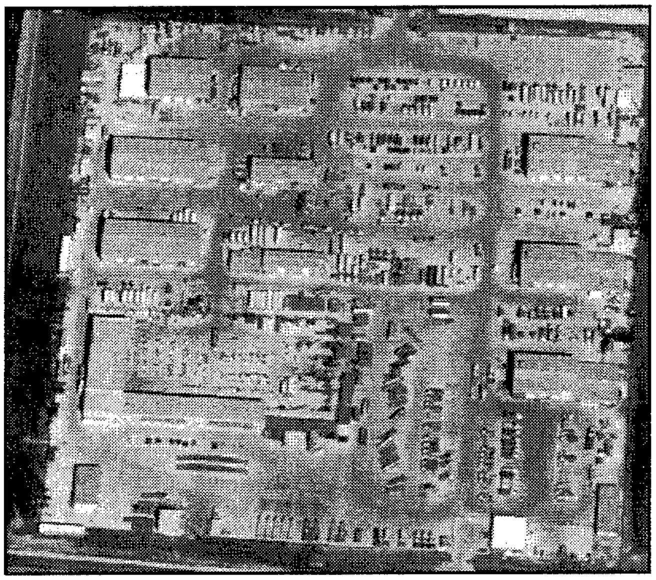


Figure B-73: DARK_FHOV425 image

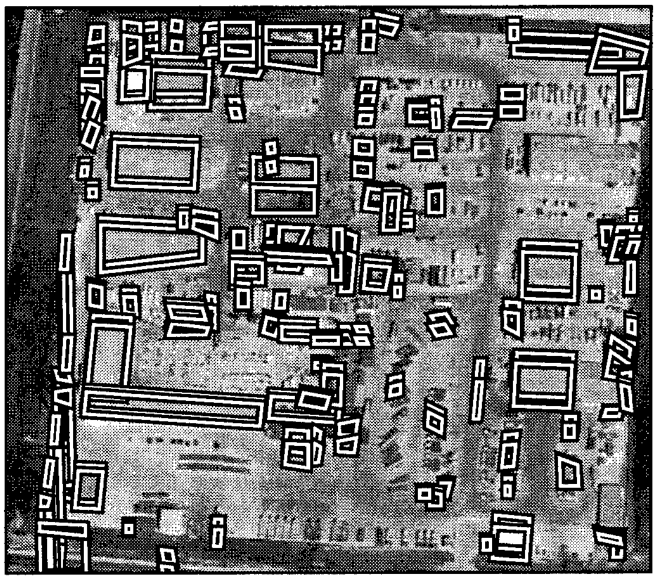


Figure B-74: BUILD+SHAVE results, DARK_FHOV425

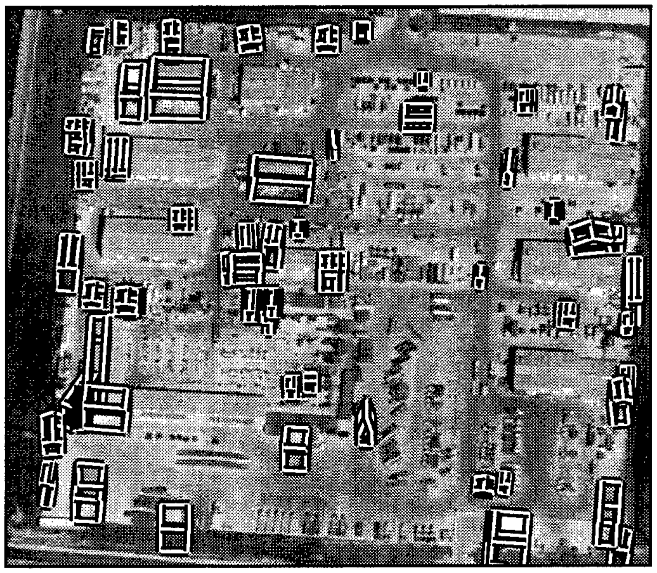


Figure B-75: VHBUILD results, DARK_FHOV425

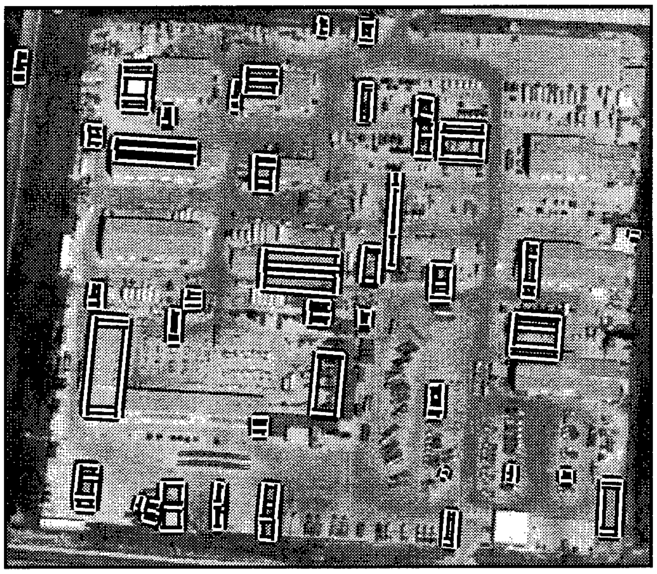


Figure B-76: PIVOT results, DARK_FHOV425

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	57.50	**1.030**	**36.12**	-	-	-
BUILD+SHAVE	**65.58**	1.699	31.02	**39.35**	3.753	**15.89**
VHBUILD	31.24	2.022	19.15	13.92	8.673	6.31
PIVOT	29.24	1.478	20.41	18.20	**3.298**	11.37

Table B-23: 2D and 3D evaluation results for DARK_FHOV425

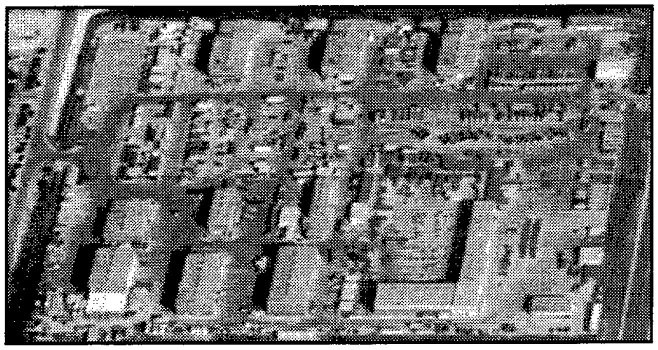


Figure B-77: DARK_FHOV527 image

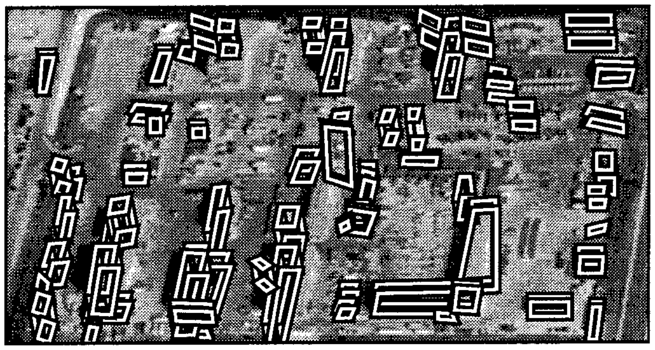


Figure B-78: BUILD+SHAVE results, DARK_FHOV527

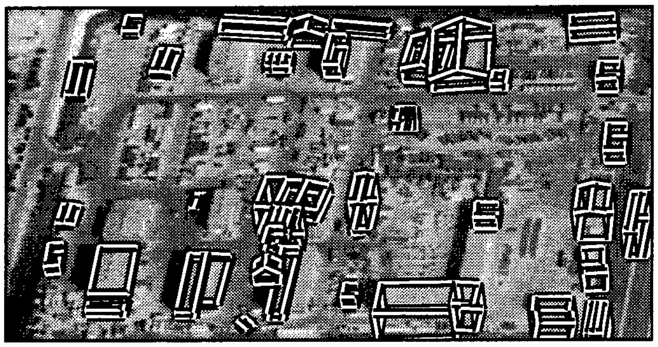


Figure B-79: VHBUILD results, DARK_FHOV527

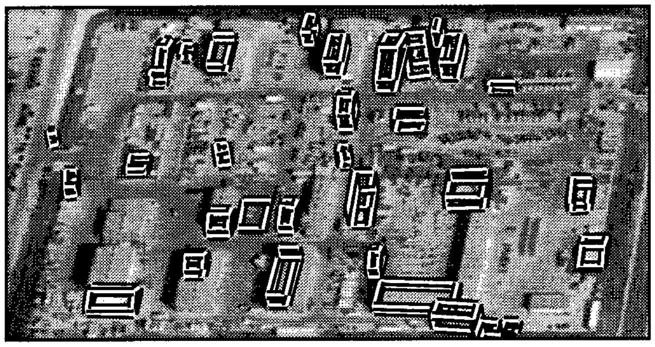


Figure B-80: PIVOT results, DARK_FHOV527

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	52.17	**0.734**	37.72	-	-	-
BUILD+SHAVE	**70.82**	1.173	**38.69**	**39.00**	2.934	**18.19**
VHBUILD	63.03	1.434	33.11	29.79	4.861	12.17
PIVOT	32.68	1.479	22.03	18.20	**2.577**	12.39

Table B-24: 2D and 3D evaluation results for DARK_FHOV527

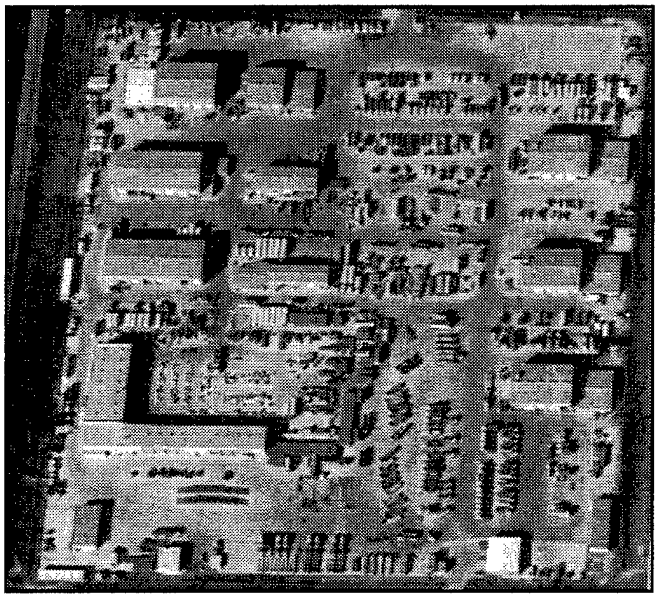


Figure B-81: DARK_FHOV727 image

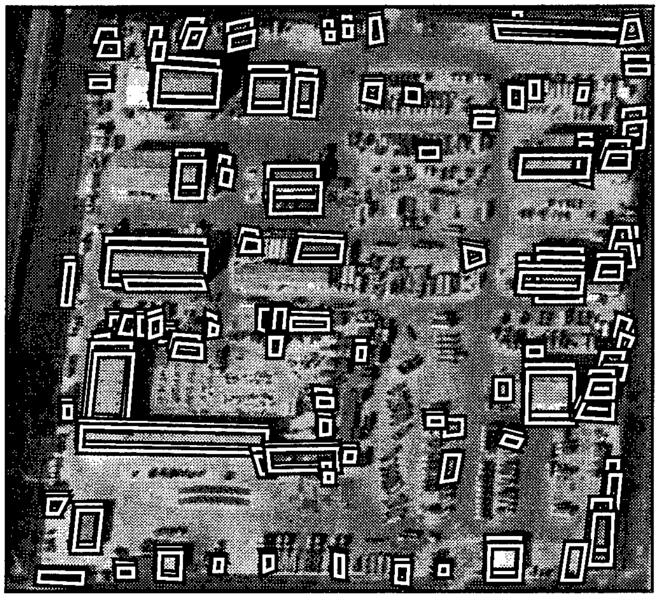


Figure B-82: BUILD+SHAVE results, DARK_FHOV727

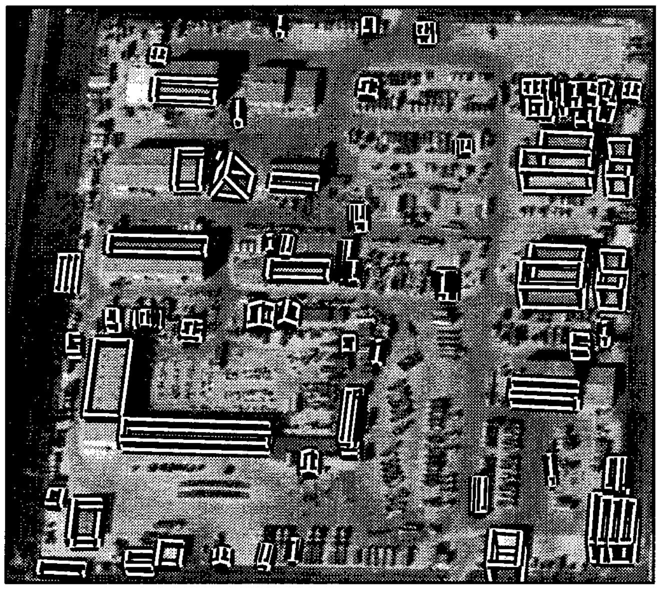


Figure B-83: VHBUILD results, DARK_FHOV727

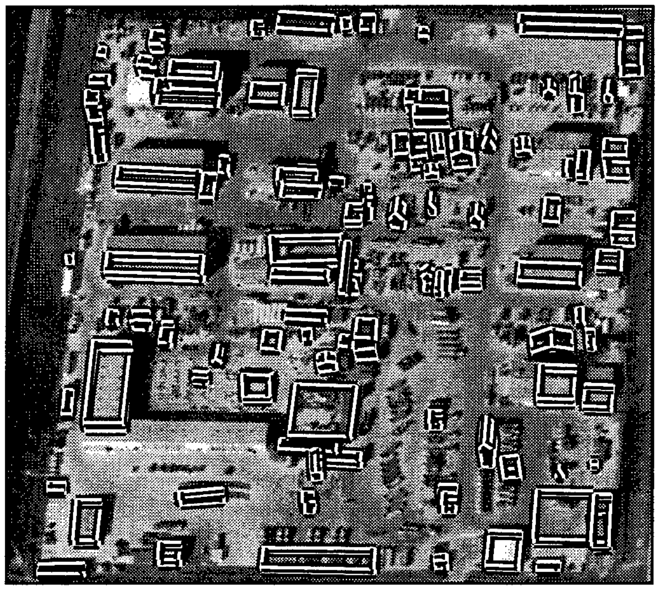


Figure B-84: PIVOT results, DARK_FHOV727

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	63.45	**0.624**	45.44	-	-	-
BUILD+SHAVE	**74.06**	0.837	**45.72**	**60.01**	**1.082**	**36.39**
VHBUILD	58.10	0.882	38.42	31.35	2.927	16.35
PIVOT	52.27	1.603	28.44	31.17	2.289	18.19

Table B-25: 2D and 3D evaluation results for DARK_FHOV727

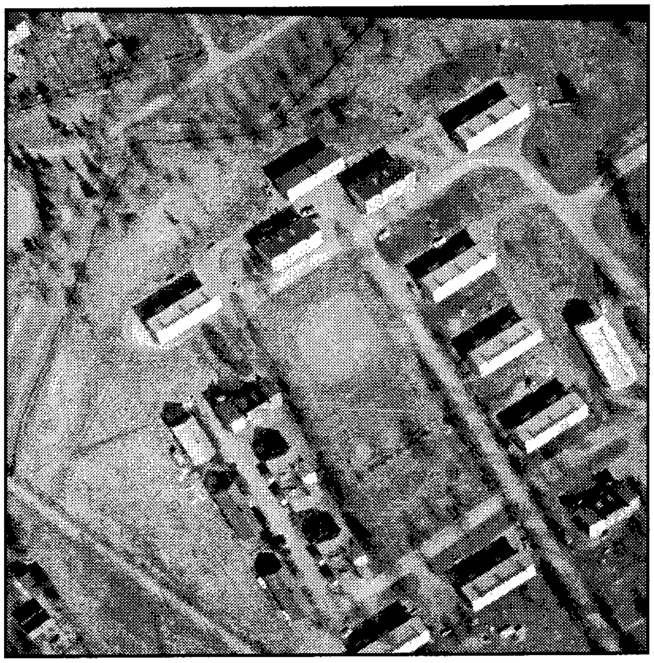


Figure B-85: FLAT_L image

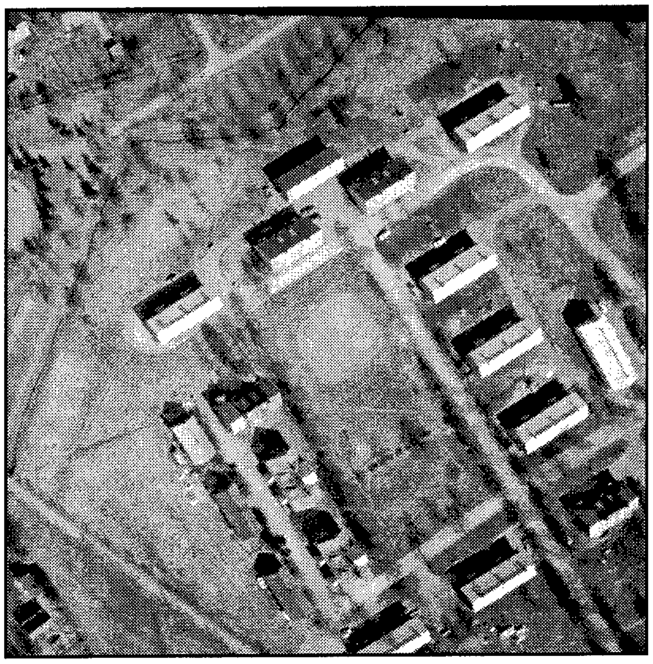


Figure B-86: BUILD+SHAVE results, FLAT_L

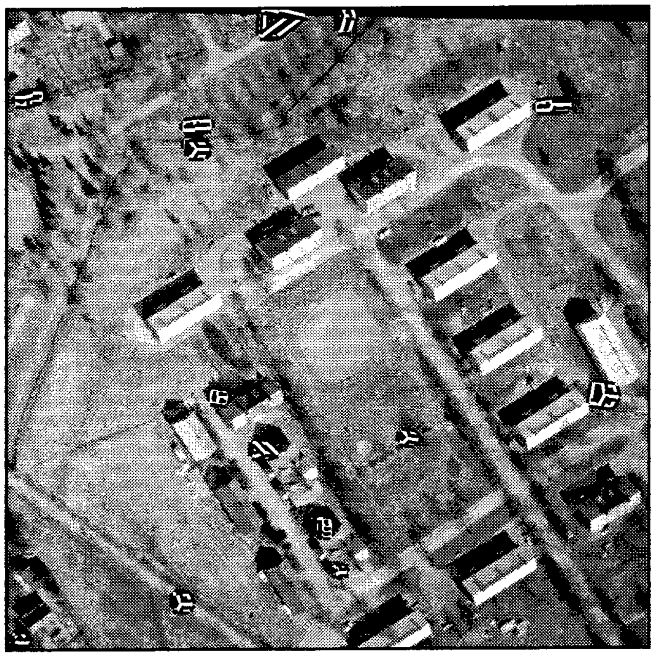


Figure B-87: VHBUILD results, FLAT_L

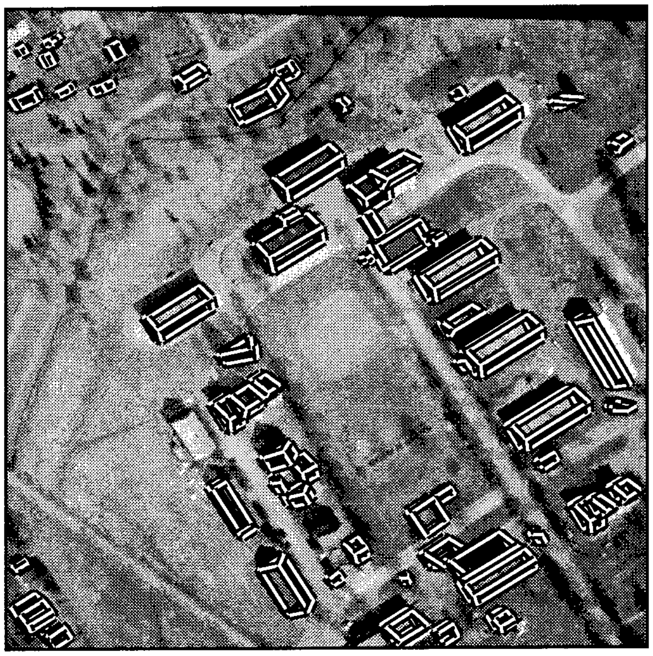


Figure B-88: PIVOT results, FLAT_L

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	0.00	**0.000**	0.00	-	-	-
BUILD+SHAVE	0.00	**0.000**	0.00	0.00	**0.000**	0.00
VHBUILD	0.65	15.021	0.60	0.14	63.530	0.13
PIVOT	**77.69**	0.625	**52.31**	**61.51**	0.944	**38.91**

Table B-26: 2D and 3D evaluation results for FLAT_L

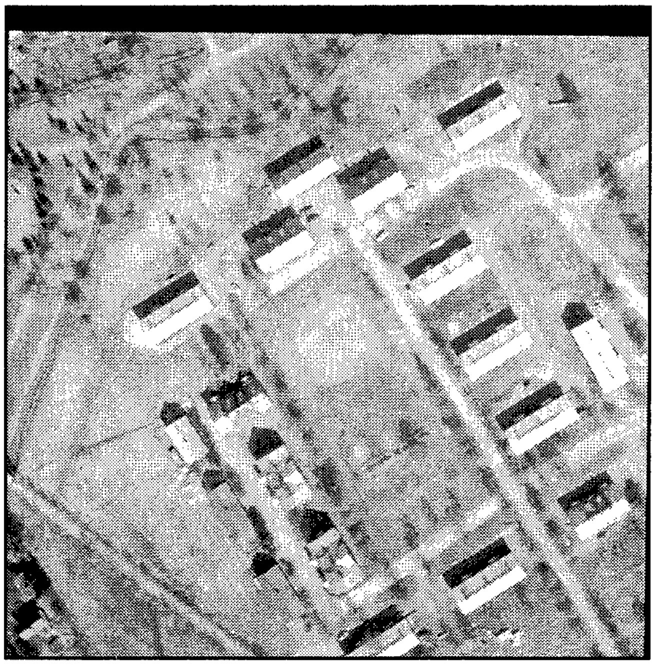


Figure B-89: FLAT_R image

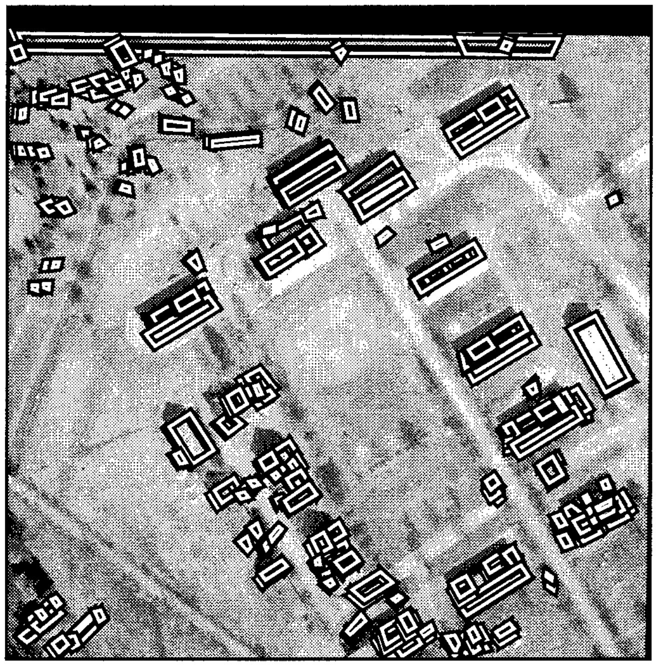


Figure B-90: BUILD+SHAVE results, FLAT_R

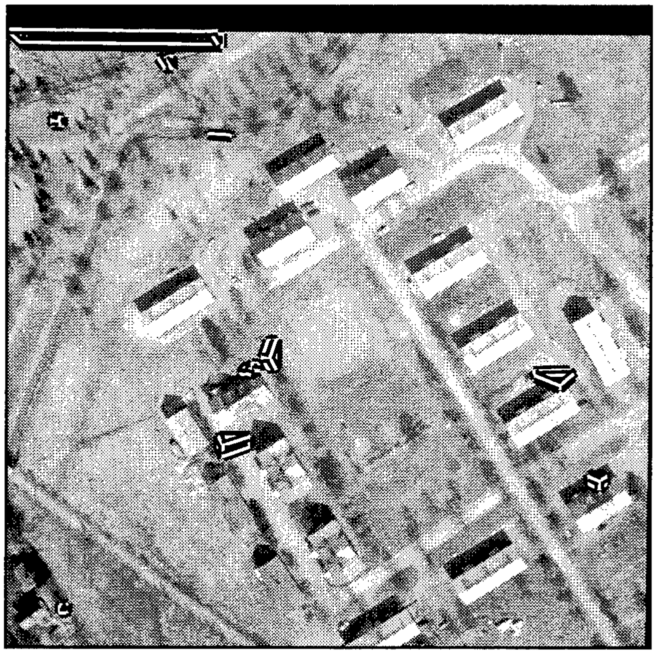


Figure B-91: VHBUILD results, FLAT_R

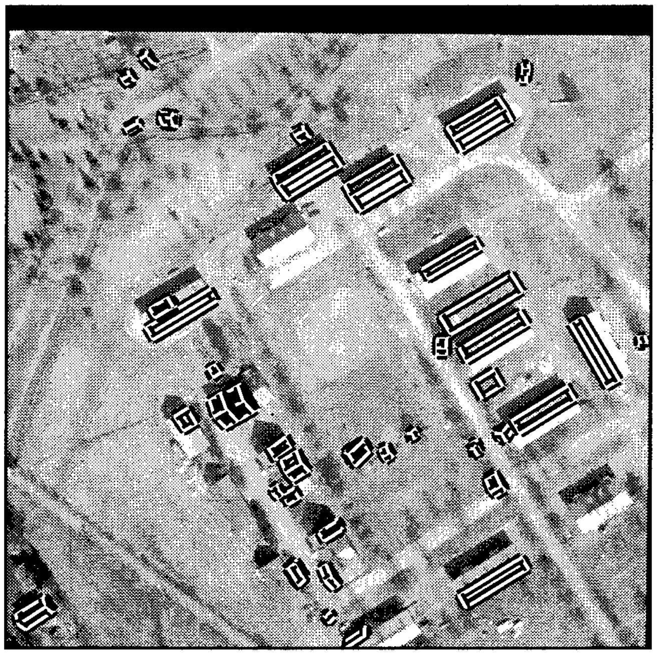


Figure B-92: PIVOT results, FLAT_R

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	75.16	0.733	**48.47**	-	-	-
BUILD+SHAVE	**78.67**	1.061	42.89	**53.50**	2.476	23.02
VHBUILD	0.15	120.892	0.12	0.00	78533	0.00
PIVOT	43.40	**0.489**	35.80	31.12	**0.949**	**24.03**

Table B-27: 2D and 3D evaluation results for FLAT_R

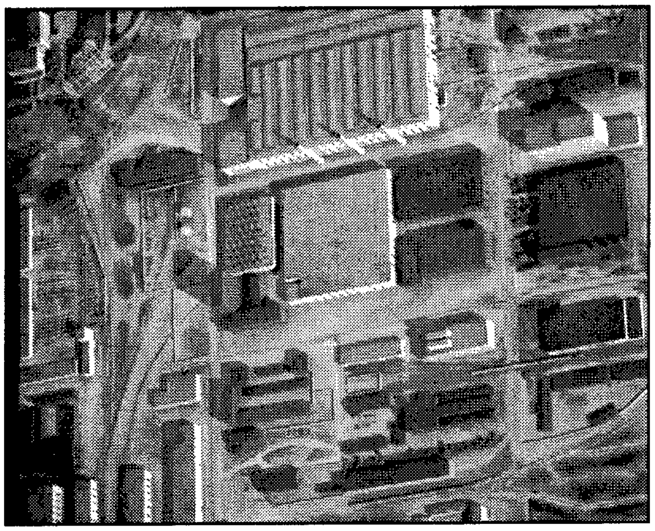


Figure B-93: J1WHOLE image

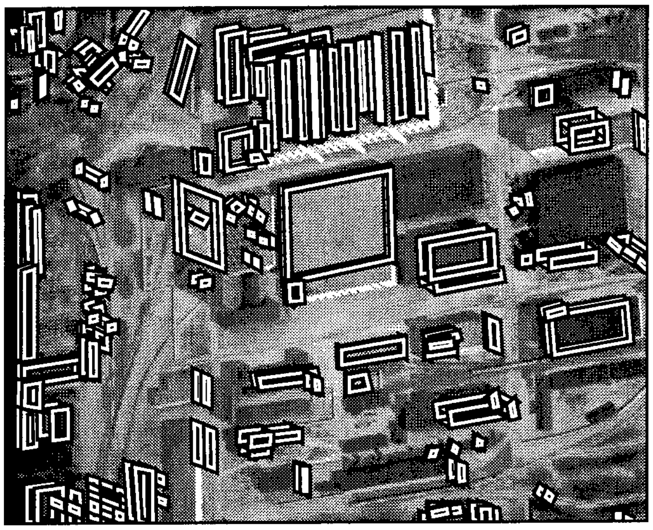


Figure B-94: BUILD+SHAVE results, J1WHOLE

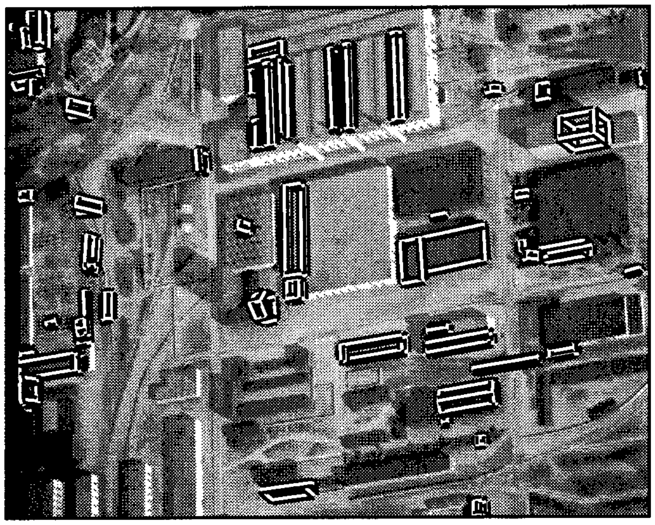


Figure B-95: VHBUILD results, J1WHOLE

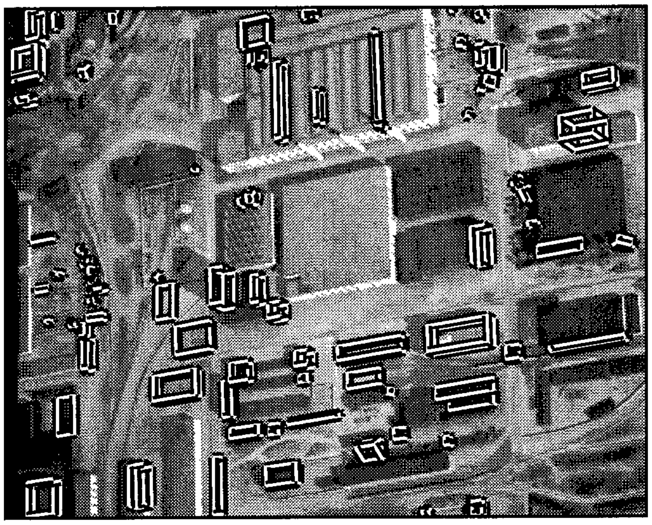


Figure B-96: PIVOT results, J1WHOLE

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	32.34	**0.227**	30.13	-	-	-
BUILD+SHAVE	**40.21**	0.418	**34.42**	**16.46**	1.093	**13.95**
VHBUILD	17.78	0.263	16.98	6.33	0.512	6.13
PIVOT	18.80	0.375	17.57	7.88	**0.511**	7.58

Table B-28: 2D and 3D evaluation results for J1WHOLE

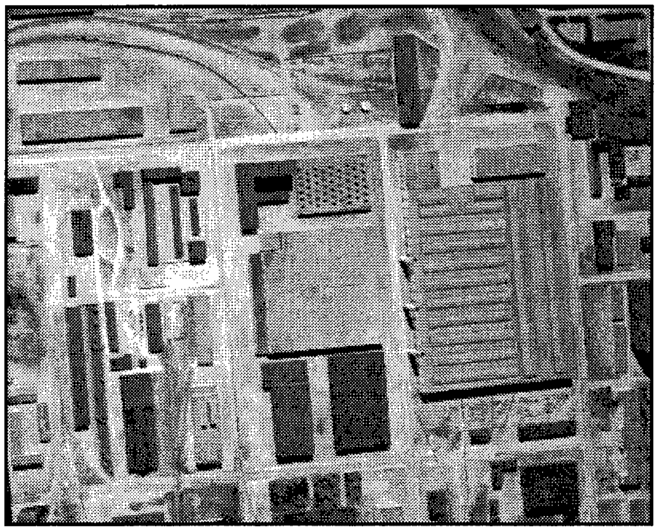


Figure B-97: J24WHOLE image

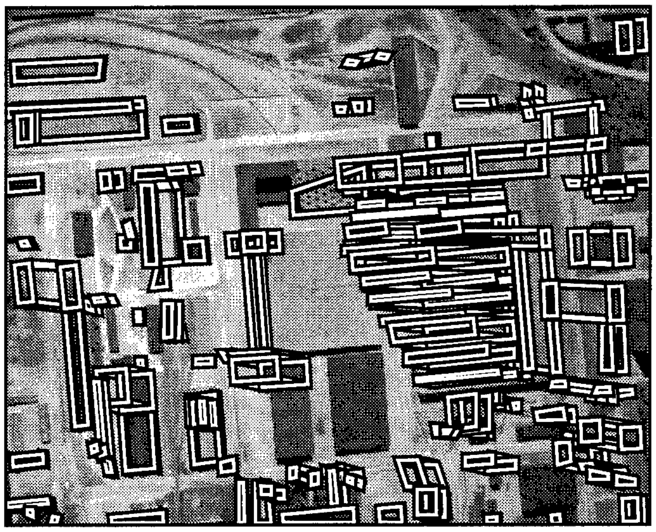


Figure B-98: BUILD+SHAVE results, J24WHOLE

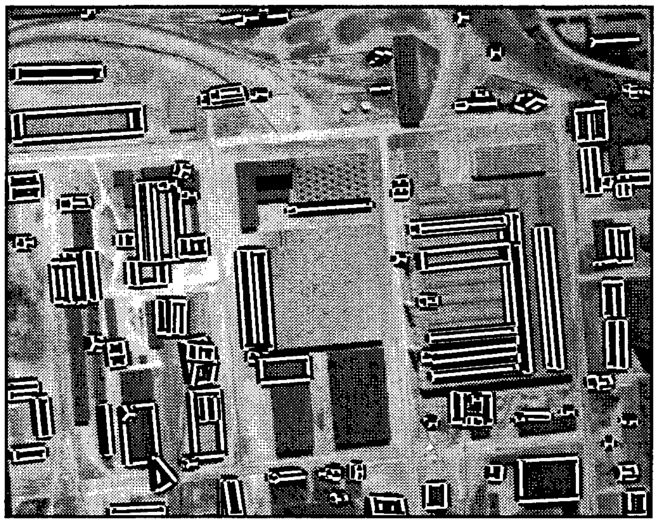


Figure B-99: VHBUILD results, J24WHOLE

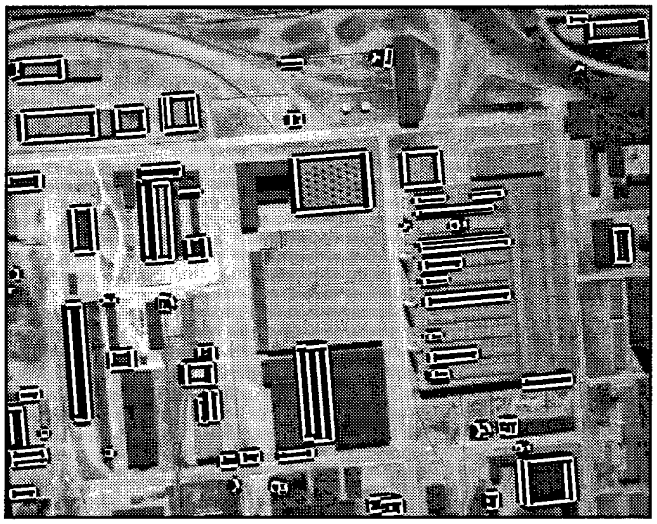


Figure B-100: PIVOT results, J24WHOLE

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	34.11	0.346	30.51	-	-	-
BUILD+SHAVE	**50.64**	0.647	**38.14**	**21.18**	9.804	6.88
VHBUILD	29.63	0.641	24.90	15.72	2.992	**10.69**
PIVOT	23.67	**0.305**	22.08	11.43	**0.754**	10.53

Table B-29: 2D and 3D evaluation results for J24WHOLE

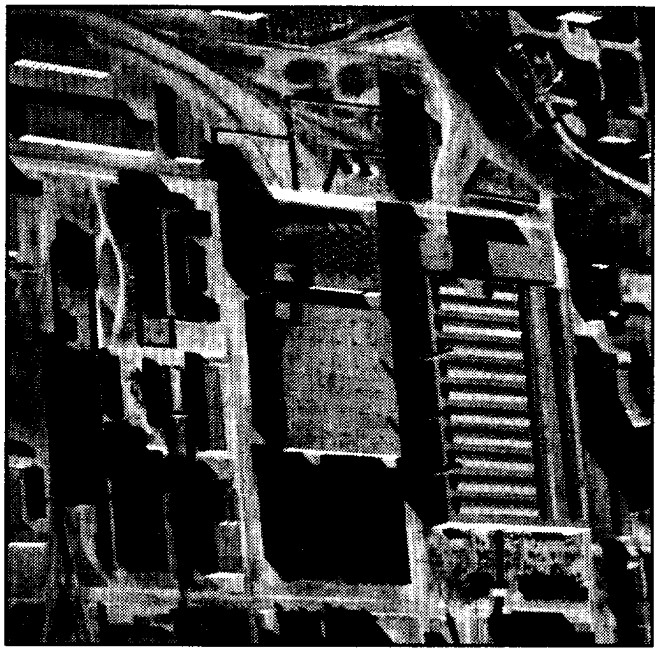


Figure B-101: J2TRIM image

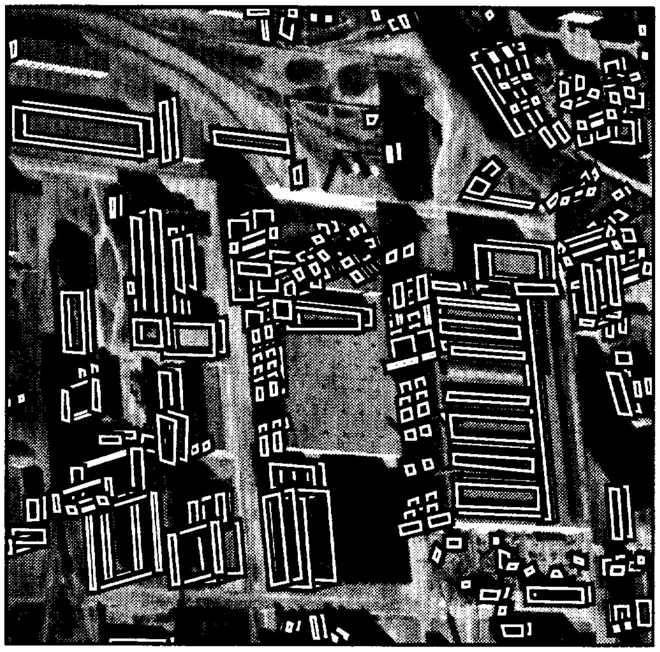


Figure B-102: BUILD+SHAVE results, J2TRIM

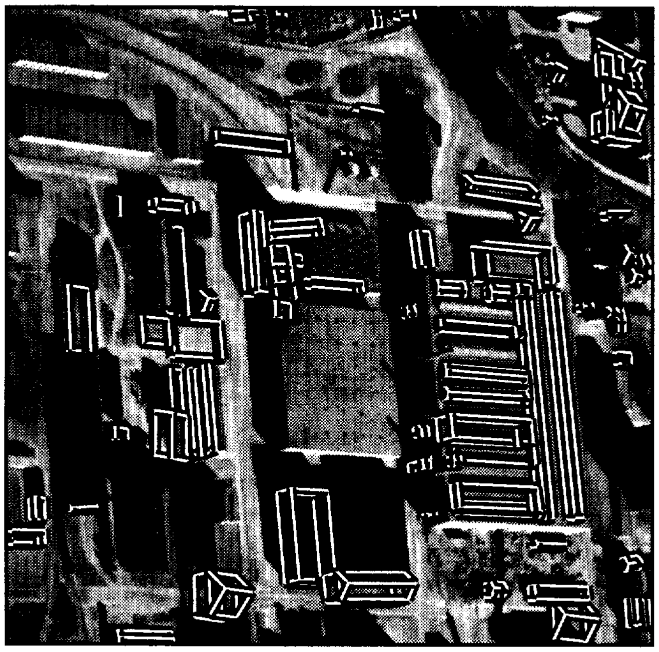


Figure B-103: VHBUILD results, J2TRIM

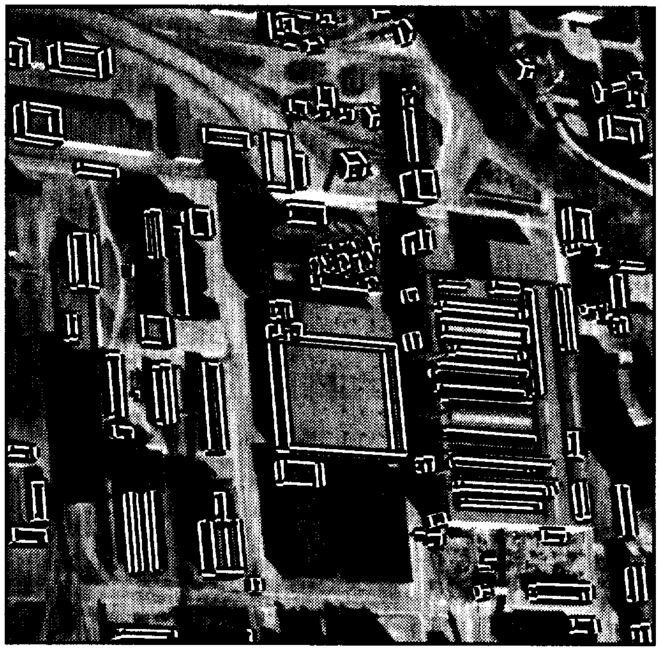


Figure B-104: PIVOT results, J2TRIM

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	31.33	0.292	28.70	-	-	-
BUILD+SHAVE	**42.92**	0.665	33.39	11.75	2.603	9.00
VHBUILD	28.76	0.533	24.94	12.83	0.856	11.56
PIVOT	38.82	**0.196**	**36.08**	**18.24**	**0.197**	**17.61**

Table B-30: 2D and 3D evaluation results for J2TRIM

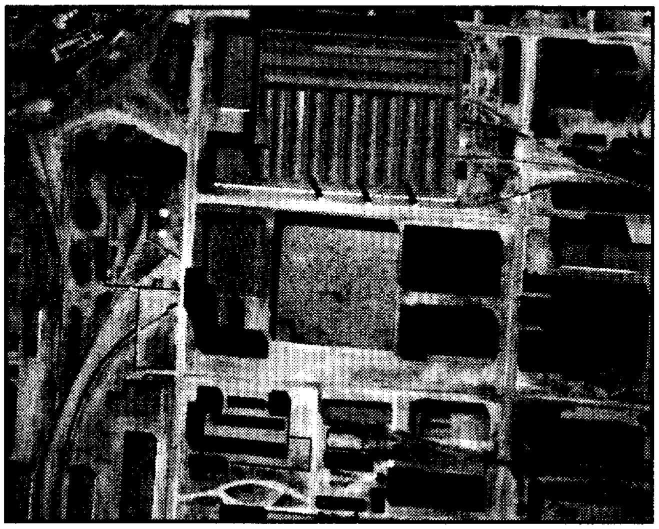


Figure B-105: J3WHOLE image

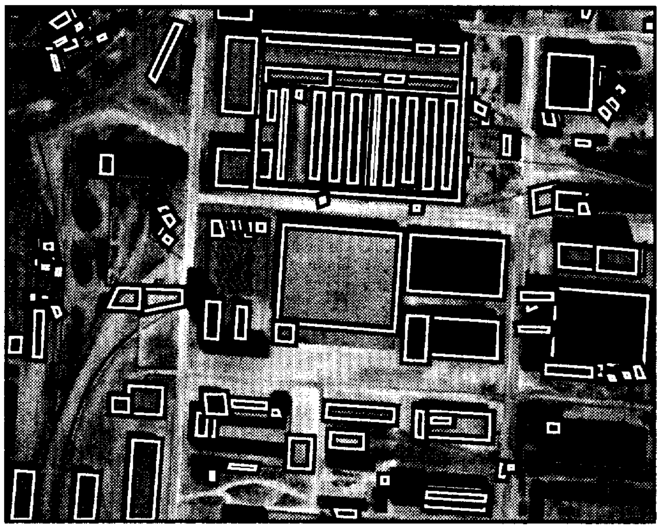


Figure B-106: BUILD+SHAVE results, J3WHOLE

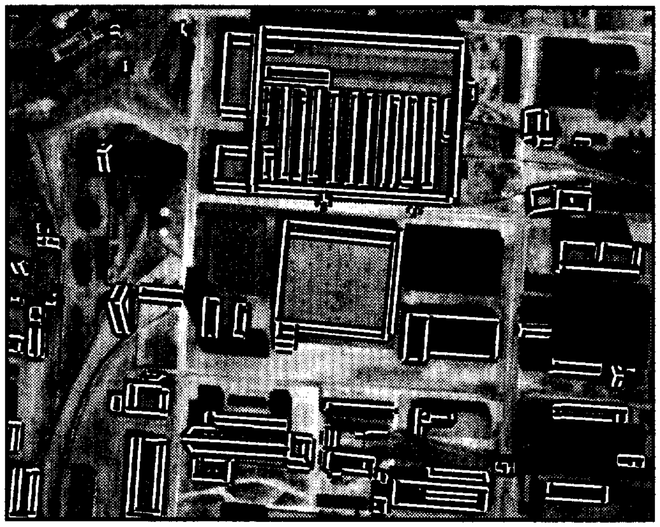


Figure B-107: VHBUILD results, J3WHOLE

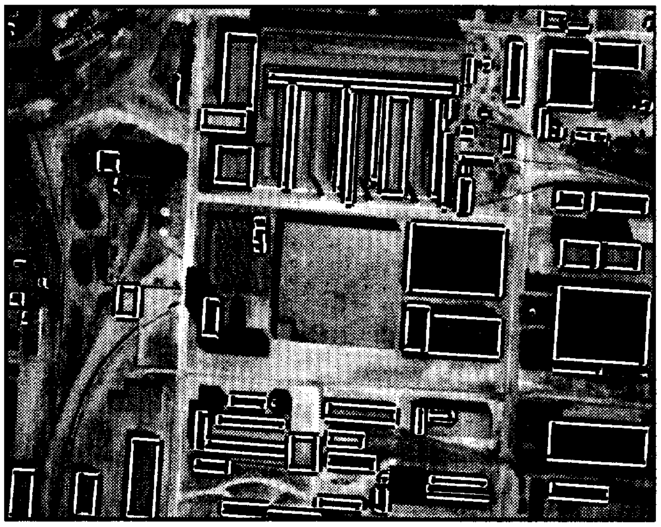


Figure B-108: PIVOT results, J3WHOLE

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	75.37	**0.093**	70.42	-	-	-
BUILD+SHAVE	**77.19**	0.120	**70.67**	**83.59**	0.562	**56.86**
VHBUILD	65.16	0.190	58.00	68.06	5.674	14.00
PIVOT	47.80	0.169	44.23	30.74	**0.530**	26.43

Table B-31: 2D and 3D evaluation results for J3WHOLE

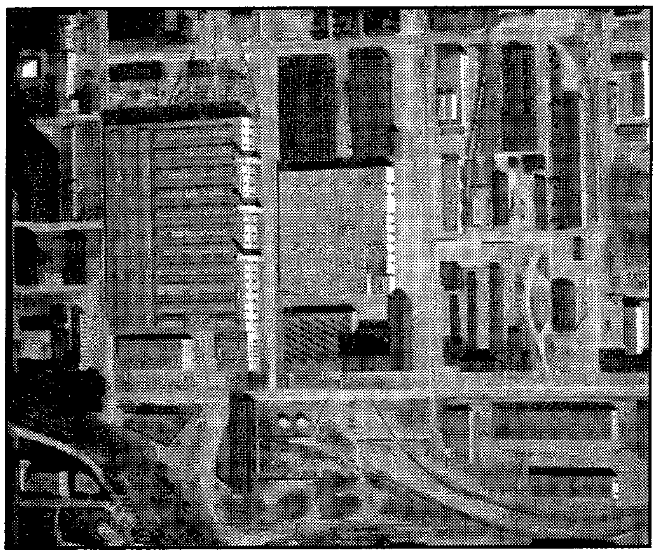


Figure B-109: J4TRIM image

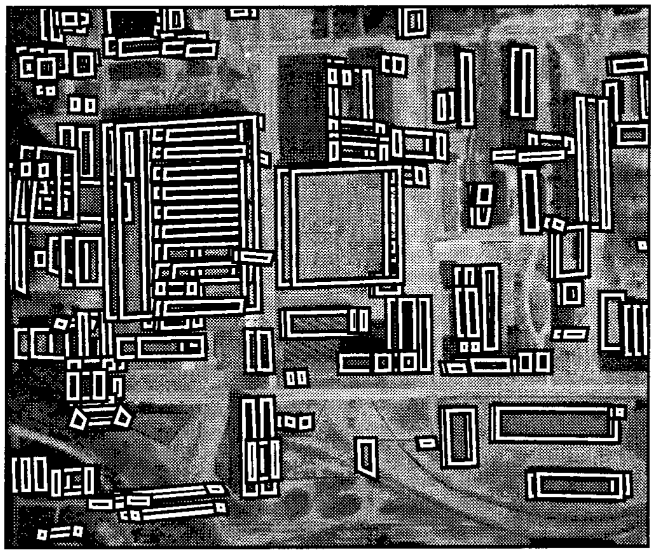


Figure B-110: BUILD+SHAVE results, J4TRIM

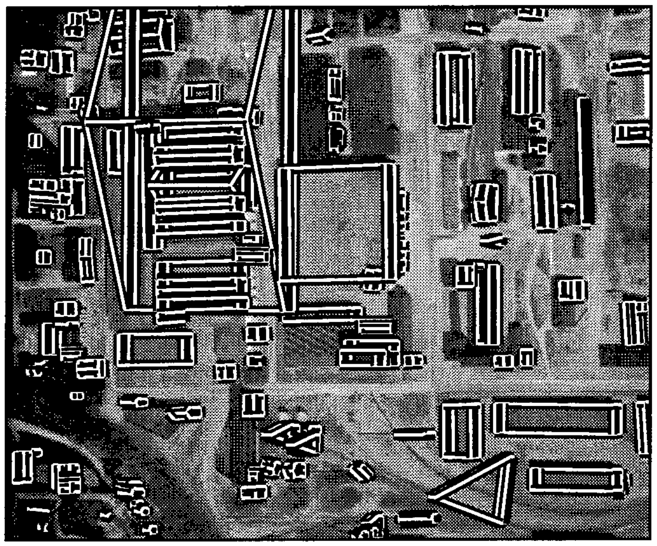


Figure B-111: VHBUILD results, J4TRIM

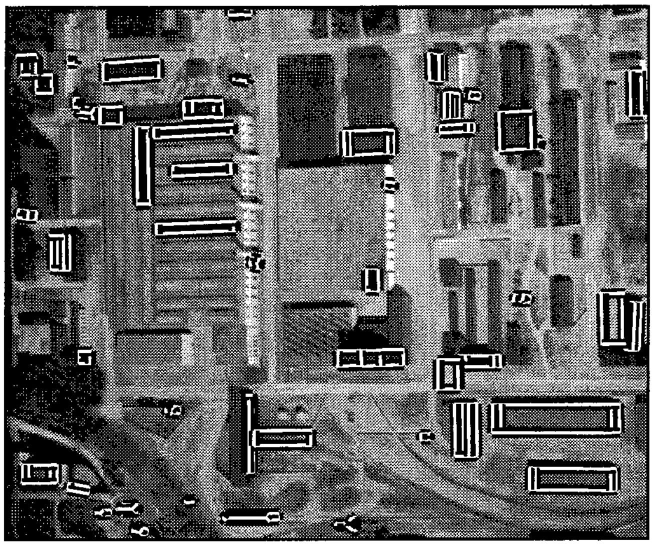


Figure B-112: PIVOT results, J4TRIM

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	61.32	**0.176**	**55.36**	-	-	-
BUILD+SHAVE	**71.29**	0.441	54.25	**64.12**	0.879	**41.00**
VHBUILD	64.22	0.577	46.86	63.16	2.748	23.09
PIVOT	16.81	0.387	15.78	7.44	**0.627**	7.11

Table B-32: 2D and 3D evaluation results for J4TRIM

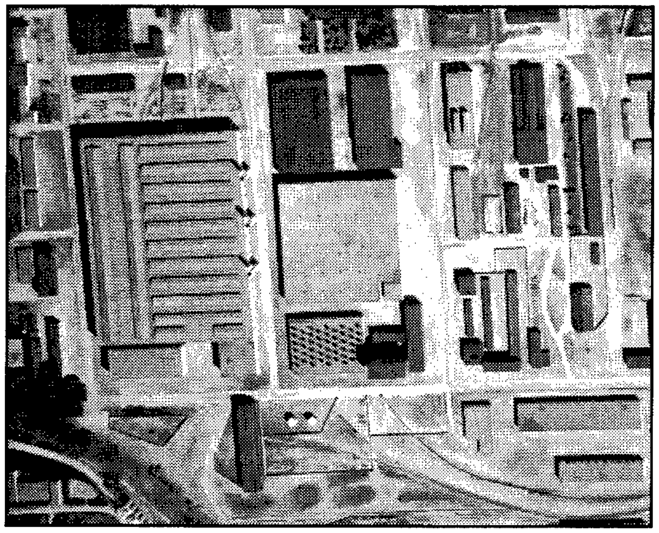


Figure B-113: J6WHOLE image

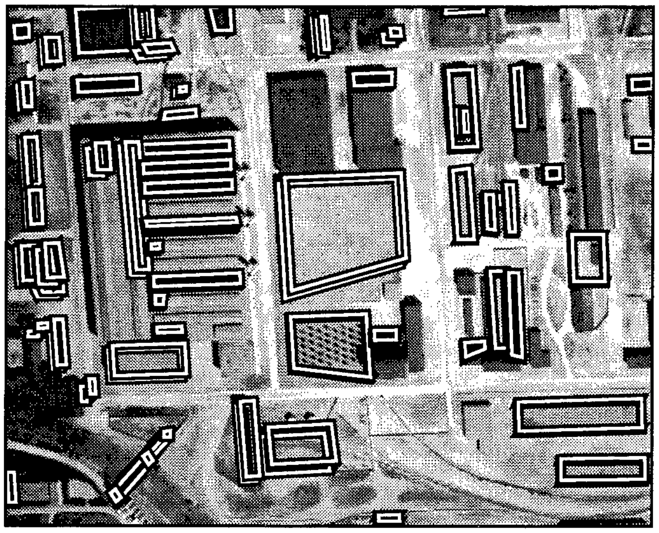


Figure B-114: BUILD+SHAVE results, J6WHOLE

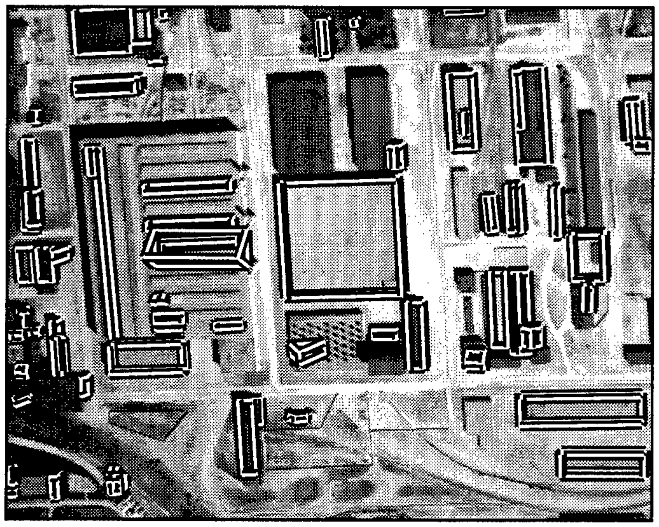


Figure B-115: VHBUILD results, J6WHOLE

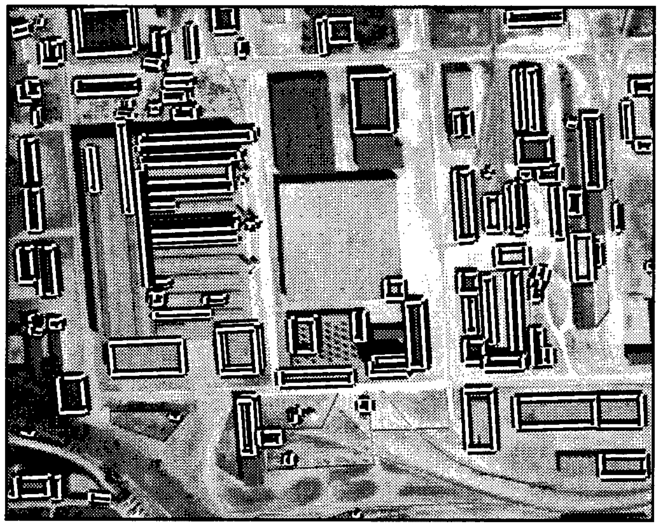


Figure B-116: PIVOT results, J6WHOLE

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	40.90	**0.182**	38.06	-	-	-
BUILD+SHAVE	**45.28**	0.283	40.13	**30.00**	0.898	23.63
VHBUILD	45.11	0.197	**41.42**	29.14	0.365	**26.33**
PIVOT	41.33	0.231	37.73	18.68	**0.301**	17.68

Table B-33: 2D and 3D evaluation results for J6WHOLE

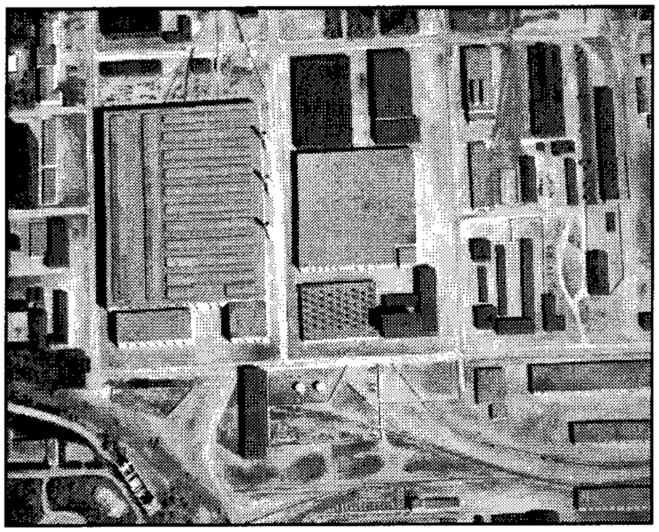


Figure B-117: J7WHOLE image

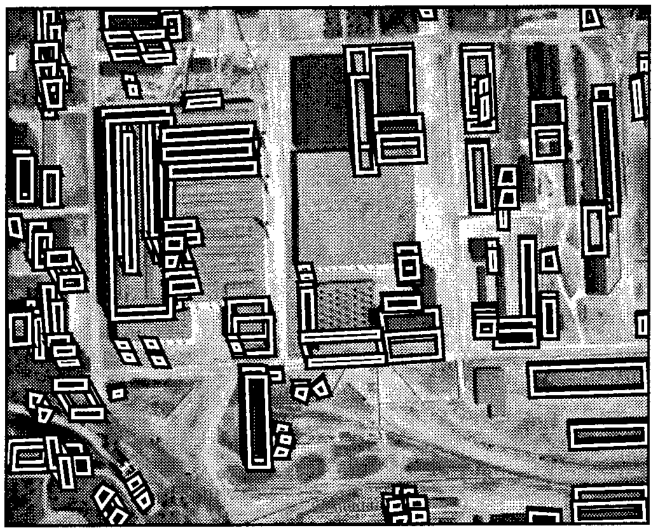


Figure B-118: BUILD+SHAVE results, J7WHOLE

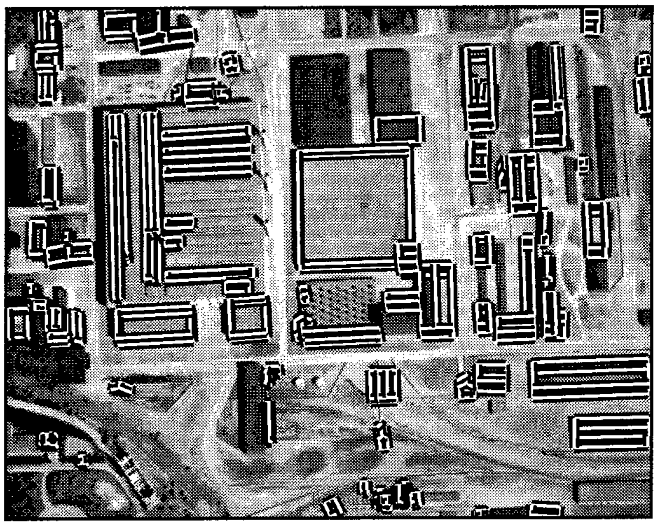


Figure B-119: VHBUILD results, J7WHOLE

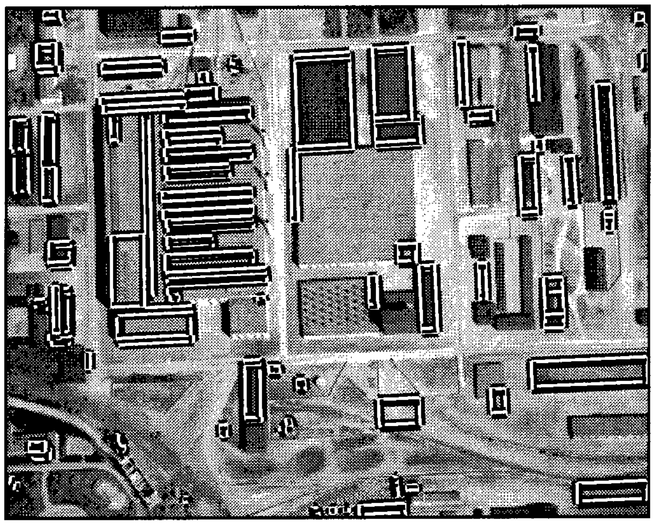


Figure B-120: PIVOT results, J7WHOLE

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.65	**0.170**	39.77	-	-	-
BUILD+SHAVE	**51.18**	0.414	**42.24**	**37.80**	2.188	20.69
VHBUILD	46.87	0.305	41.01	28.66	1.277	**20.98**
PIVOT	41.48	0.198	38.33	22.96	**0.486**	20.65

Table B-34: 2D and 3D evaluation results for J7WHOLE

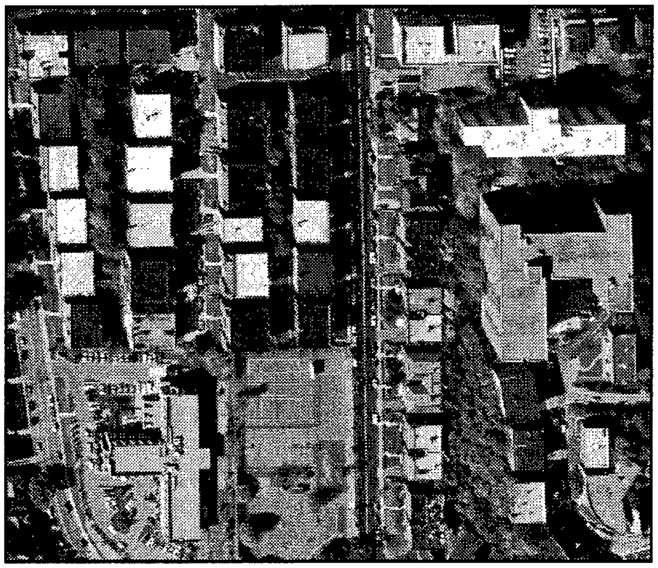


Figure B-121: MIXT_SLAG15 image

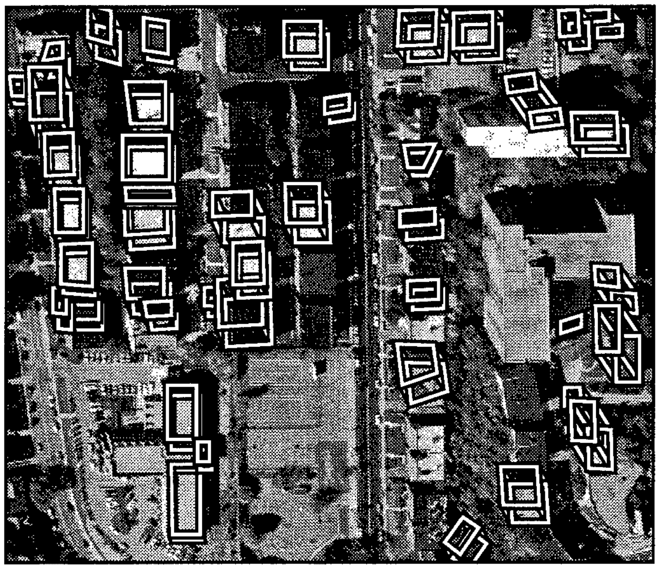


Figure B-122: BUILD+SHAVE results, MIXT_SLAG15

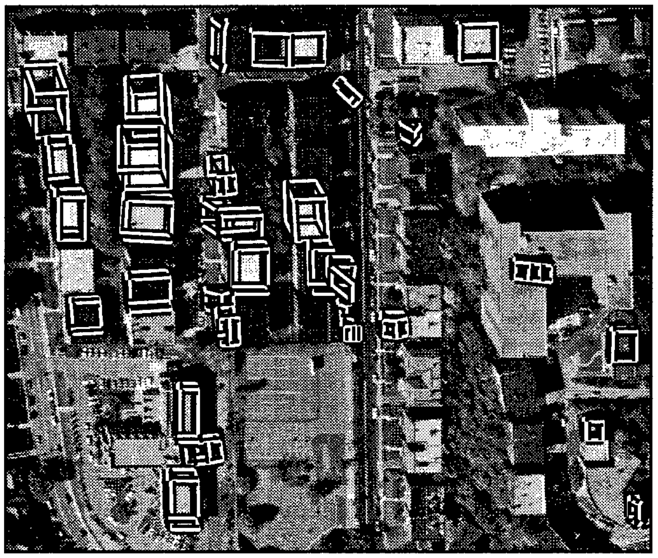


Figure B-123: VHBUILD results, MIXT_SLAG15

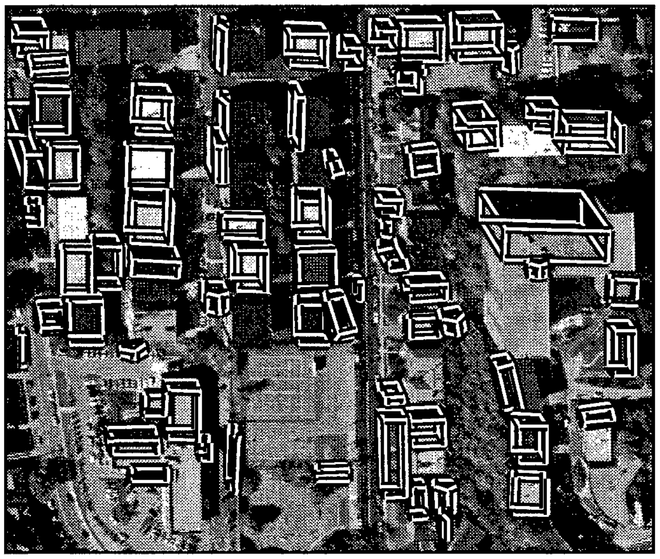


Figure B-124: PIVOT results, MIXT_SLAG15

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	30.81	**0.089**	29.98	-	-	-
BUILD+SHAVE	43.80	0.216	40.02	23.99	1.113	18.93
VHBUILD	26.41	0.231	24.90	12.10	1.021	10.77
PIVOT	**49.28**	0.358	**41.90**	**28.31**	**0.834**	**22.90**

Table B-35: 2D and 3D evaluation results for MIXT_SLAG15

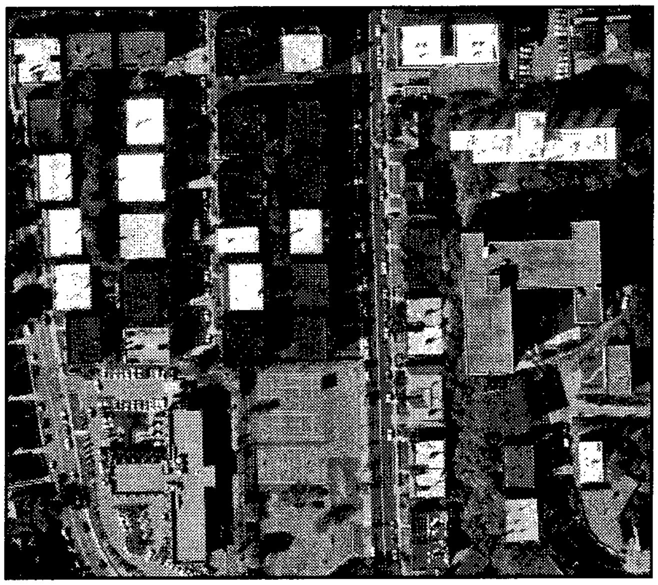


Figure B-125: MIXT_SLAG16 image

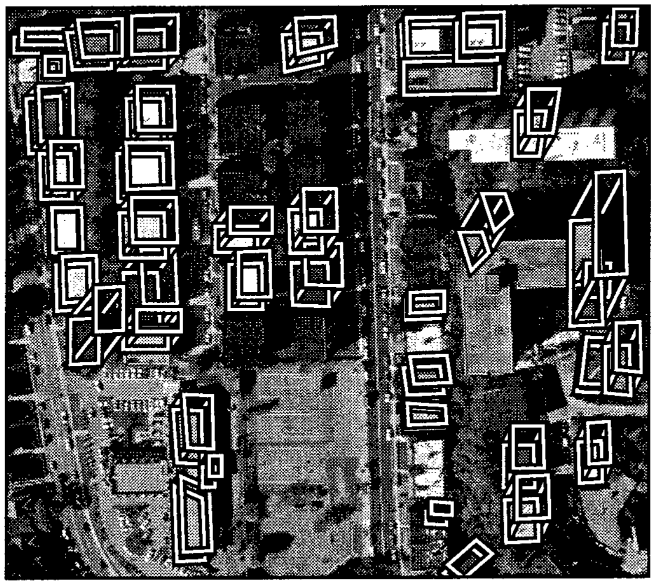


Figure B-126: BUILD+SHAVE results, MIXT_SLAG16

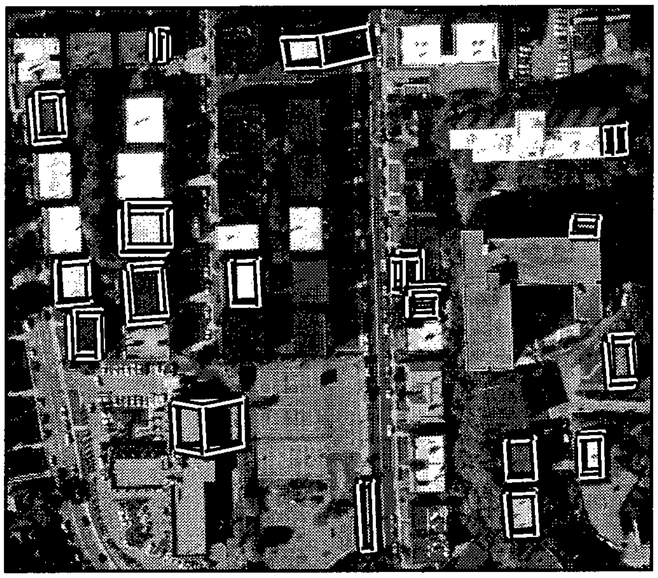


Figure B-127: VHBUILD results, MIXT_SLAG16

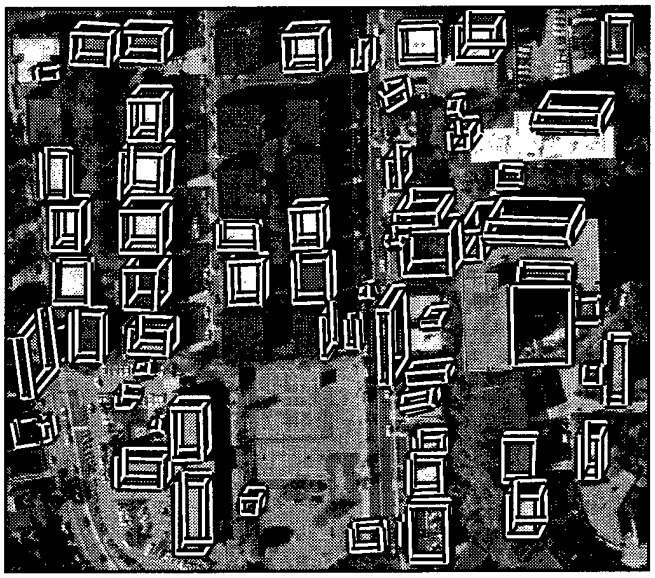


Figure B-128: PIVOT results, MIXT_SLAG16

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	33.71	**0.127**	32.32	-	-	-
BUILD+SHAVE	50.77	0.204	**46.01**	**27.50**	1.245	20.49
VHBUILD	17.44	0.195	16.87	6.13	**0.481**	5.95
PIVOT	**50.89**	0.304	44.08	25.84	0.749	**21.65**

Table B-36: 2D and 3D evaluation results for MIXT_SLAG16

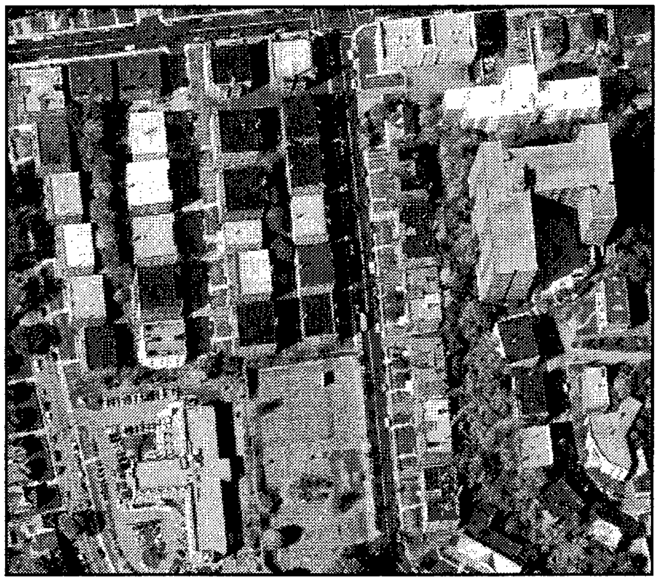


Figure B-129: MIXT_SLAG25 image

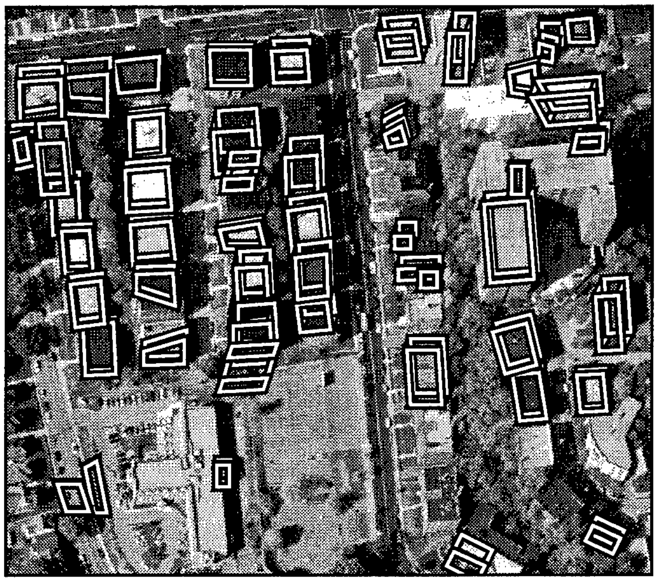


Figure B-130: BUILD+SHAVE results, MIXT_SLAG25

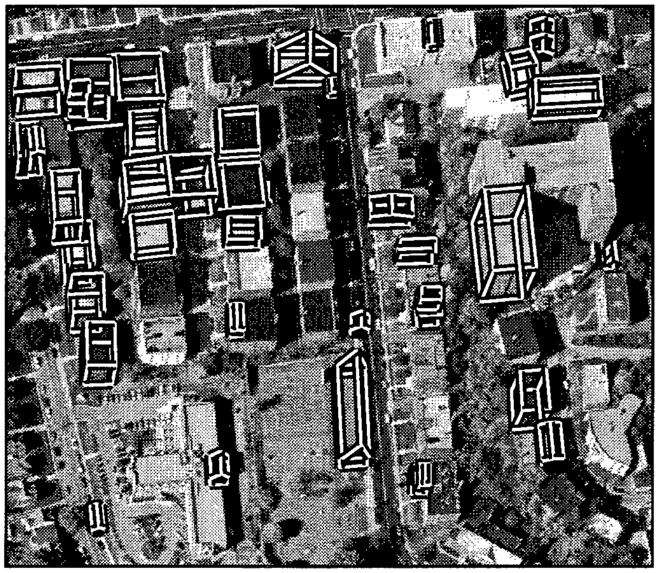


Figure B-131: VHBUILD results, MIXT_SLAG25

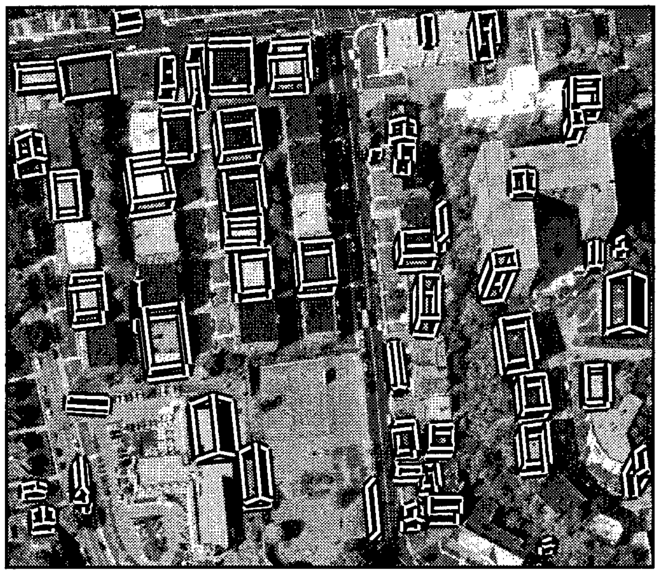


Figure B-132: PIVOT results, MIXT_SLAG25

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.51	**0.113**	40.56	-	-	-
BUILD+SHAVE	**53.87**	0.176	**49.20**	**26.52**	**0.820**	**21.79**
VHBUILD	35.99	0.385	31.61	21.39	1.517	16.15
PIVOT	37.58	0.452	32.13	19.24	0.862	16.50

Table B-37: 2D and 3D evaluation results for MIXT_SLAG25

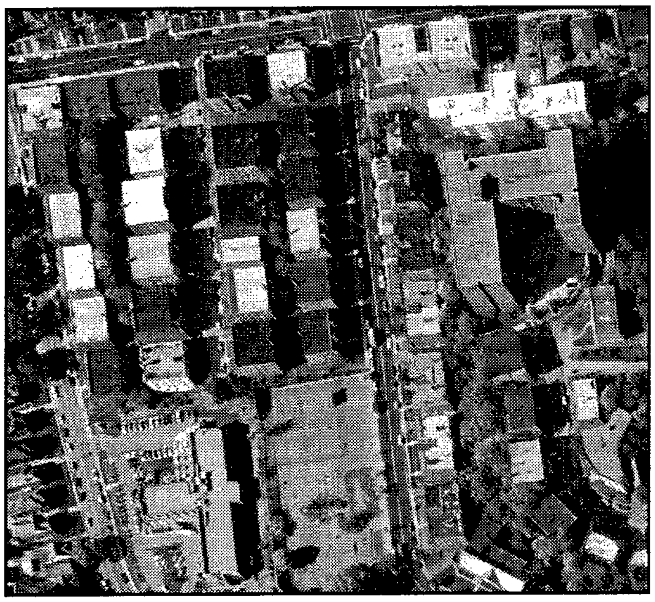


Figure B-133: MIXT_SLAG26 image

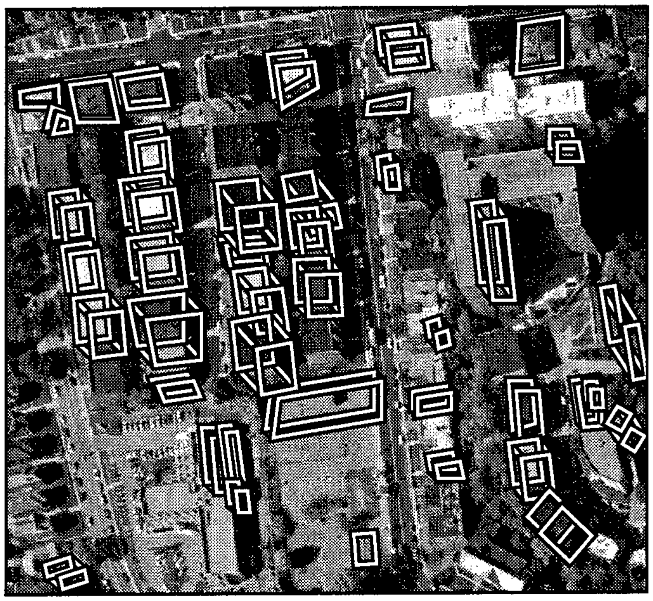


Figure B-134: BUILD+SHAVE results, MIXT_SLAG26

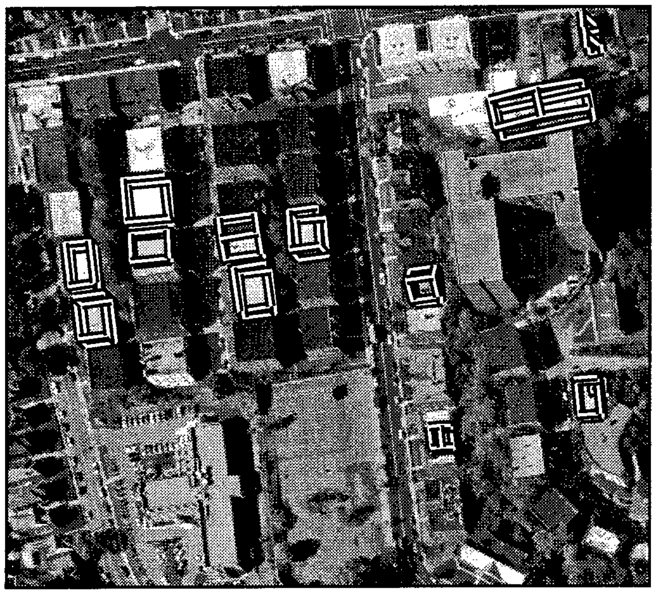


Figure B-135: VHBUILD results, MIXT_SLAG26

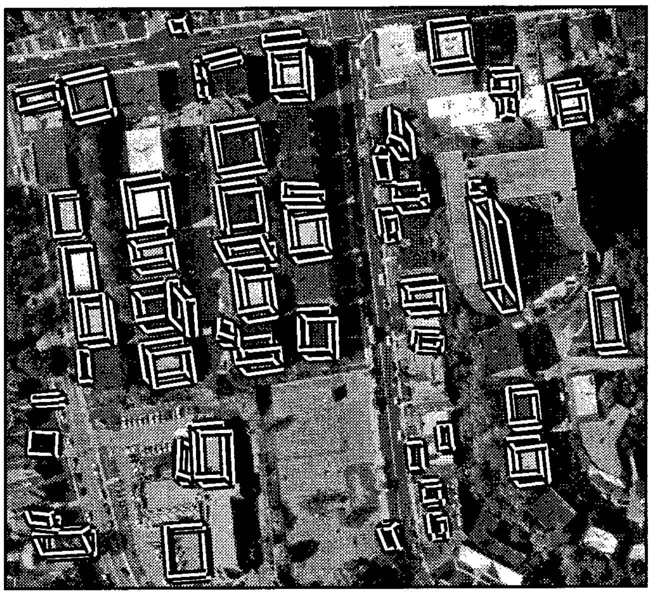


Figure B-136: PIVOT results, MIXT_SLAG26

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	29.30	0.230	27.45	-	-	-
BUILD+SHAVE	**43.28**	0.317	**38.06**	**22.08**	1.204	17.44
VHBUILD	14.83	**0.036**	14.75	5.71	0.468	5.56
PIVOT	41.28	0.210	37.98	21.38	**0.434**	**19.56**

Table B-38: 2D and 3D evaluation results for MIXT_SLAG26

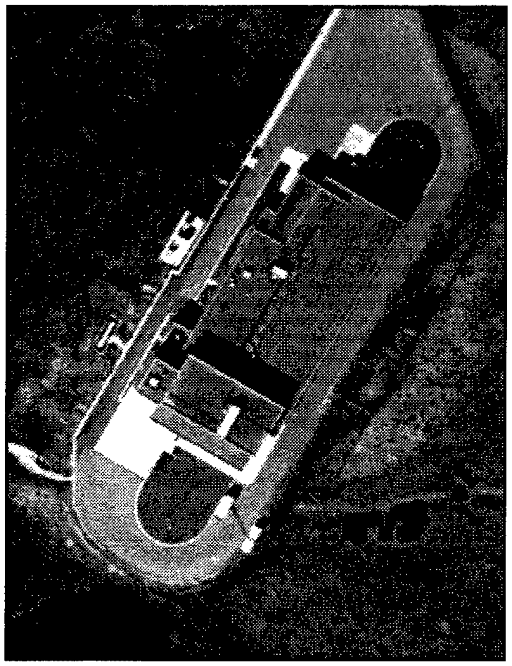


Figure B-137: MMBLD-70OV image

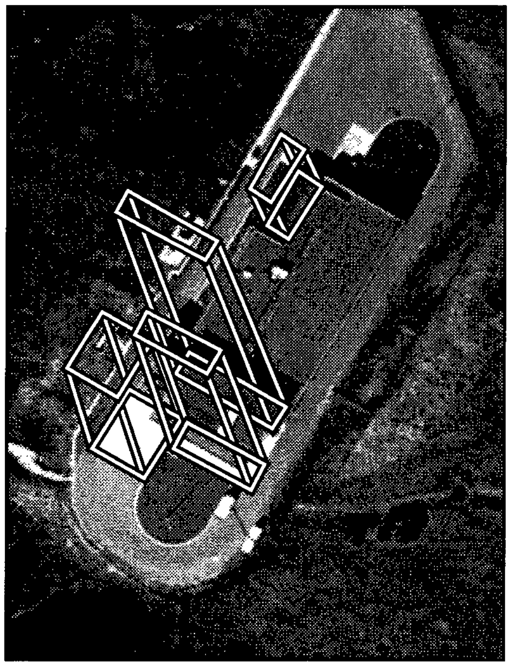


Figure B-138: BUILD+SHAVE results, MMBLD-70OV

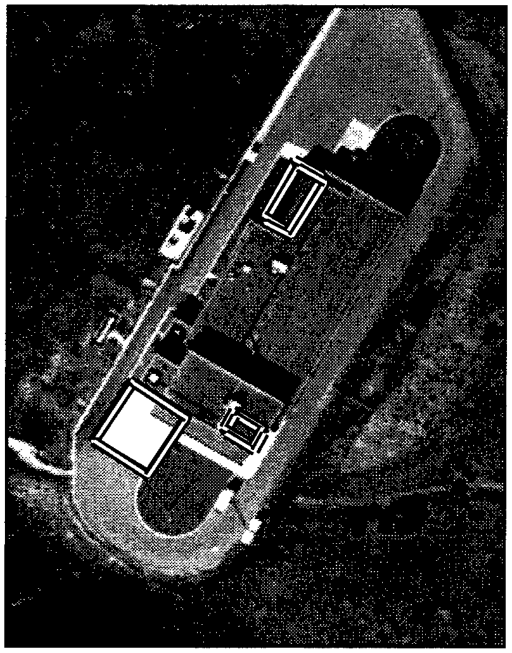


Figure B-139: VHBUILD results, MMBLD-70OV

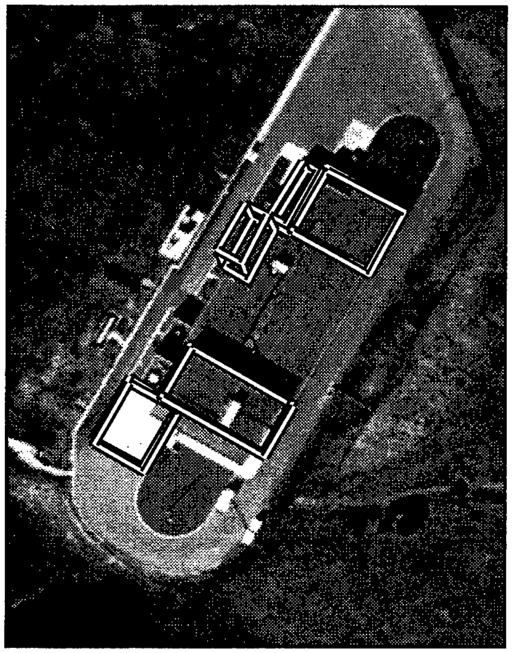


Figure B-140: PIVOT results, MMBLD-70OV

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	17.09	0.720	15.22	-	-	-
BUILD+SHAVE	**46.58**	1.259	29.36	4.60	151.405	0.58
VHBUILD	10.46	1.072	9.41	5.55	2.228	4.94
PIVOT	45.90	**0.305**	**40.27**	**34.83**	**1.334**	**23.78**

Table B-39: 2D and 3D evaluation results for MMBLD-70OV

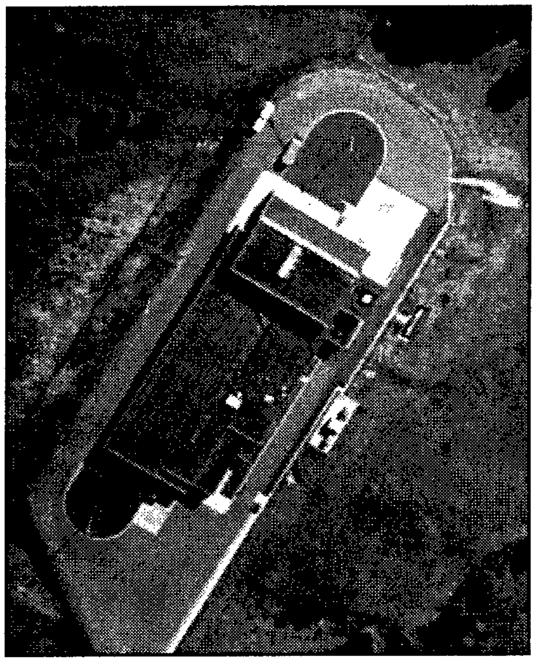


Figure B-141: MMBLD-71OV image

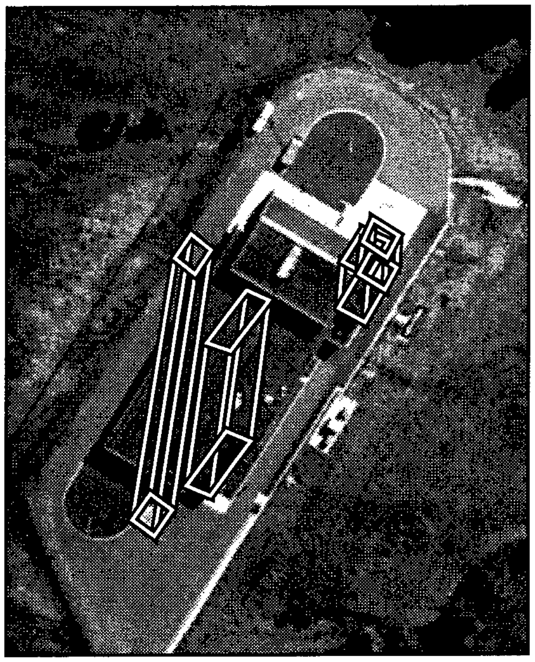


Figure B-142: BUILD+SHAVE results, MMBLD-71OV

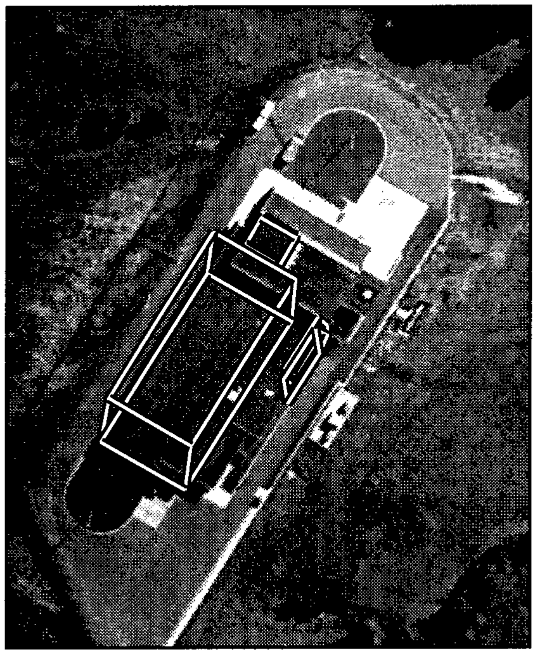


Figure B-143: VHBUILD results, MMBLD-71OV

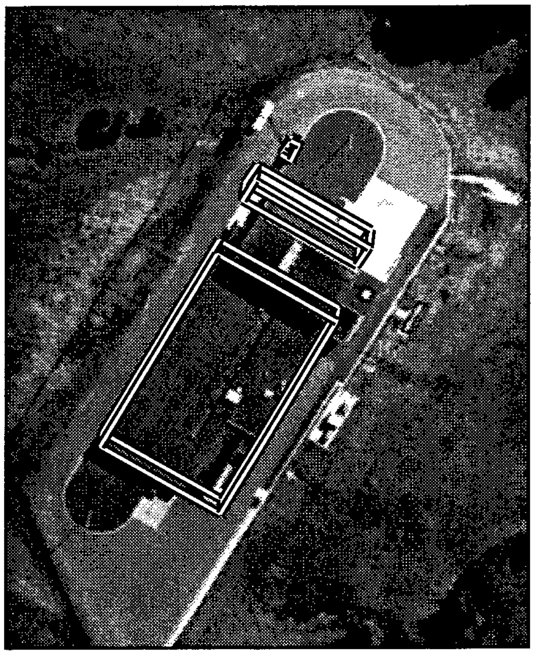


Figure B-144: PIVOT results, MMBLD-71OV

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	7.75	0.462	7.48	-	-	-
BUILD+SHAVE	45.27	0.300	39.86	4.60	44.773	1.50
VHBUILD	62.26	**0.130**	57.61	35.78	9.317	8.26
PIVOT	**81.11**	0.166	**71.48**	**60.95**	**2.816**	**22.44**

Table B-40: 2D and 3D evaluation results for MMBLD-71OV

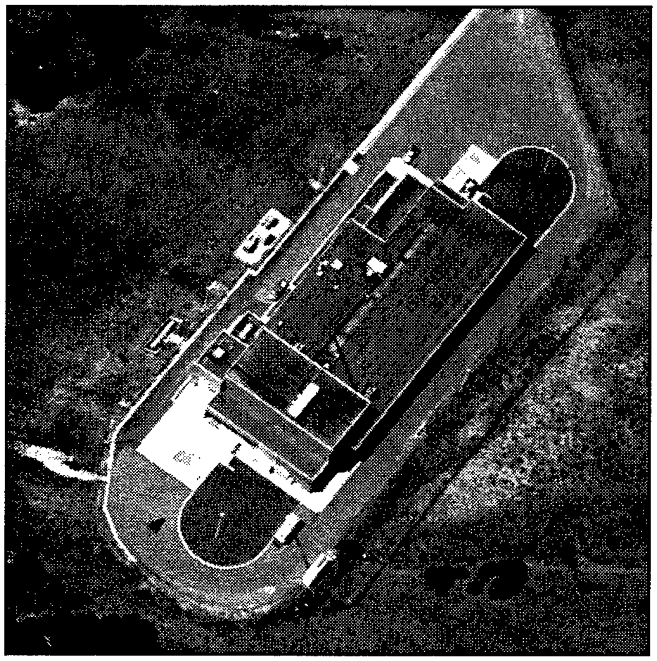


Figure B-145: MMBLD-8OV image

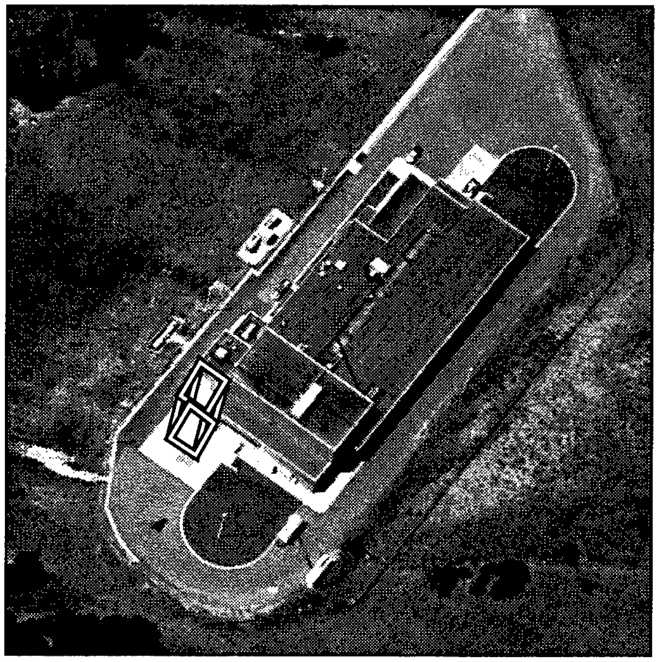


Figure B-146: BUILD+SHAVE results, MMBLD-8OV

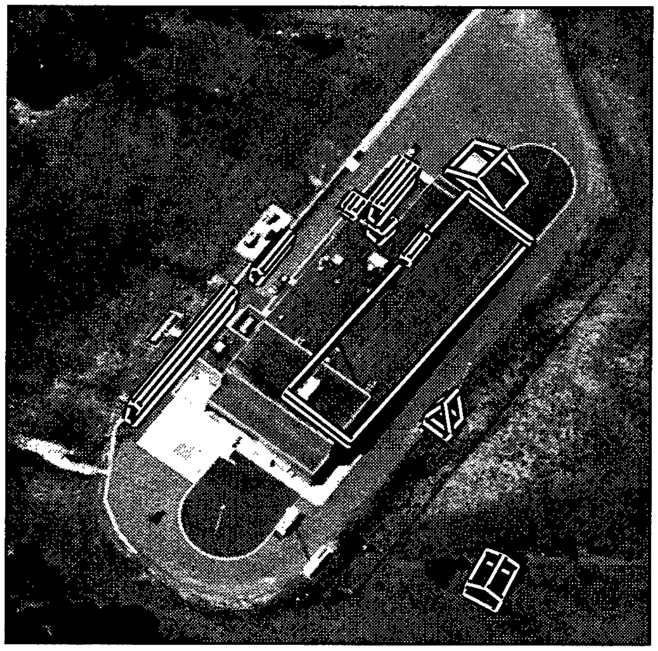


Figure B-147: VHBUILD results, MMBLD-8OV

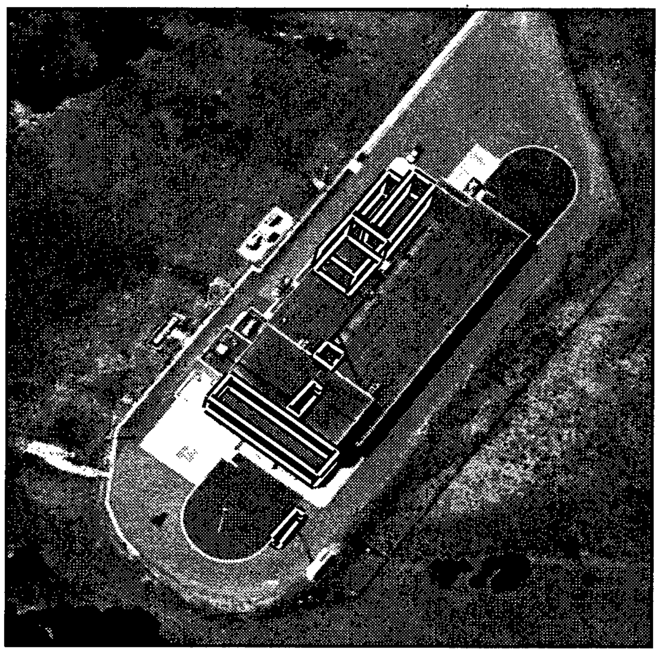


Figure B-148: PIVOT results, MMBLD-8OV

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	1.02	1.406	1.00	-	-	-
BUILD+SHAVE	1.47	2.882	1.41	0.03	328.667	0.03
VHBUILD	**48.14**	0.468	**39.29**	**27.26**	**0.777**	**22.50**
PIVOT	23.75	**0.146**	22.95	14.05	1.131	12.12

Table B-41: 2D and 3D evaluation results for MMBLD-8OV

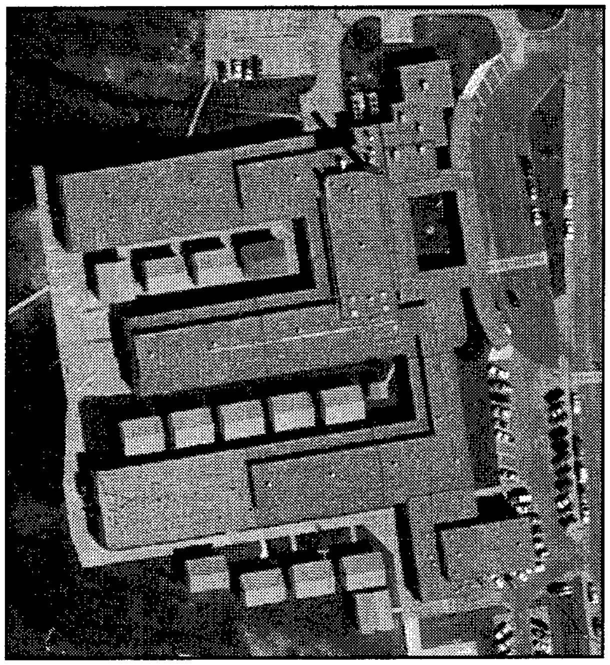


Figure B-149: NEST_FHN711 image

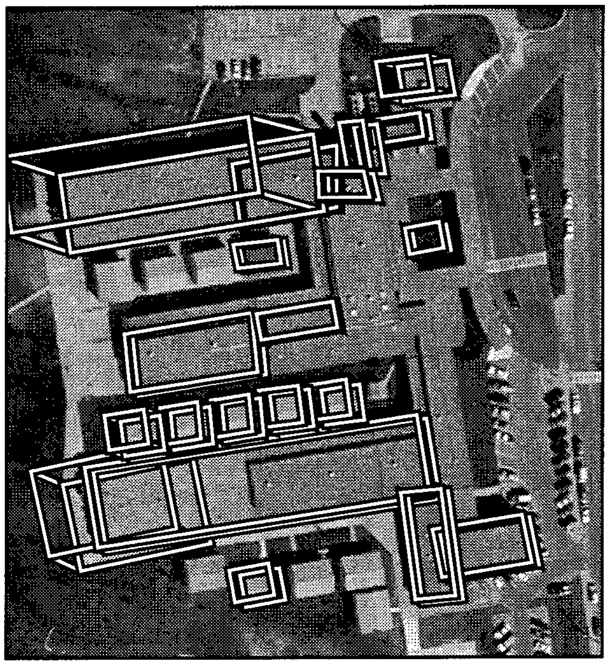


Figure B-150: BUILD+SHAVE results, NEST_FHN711

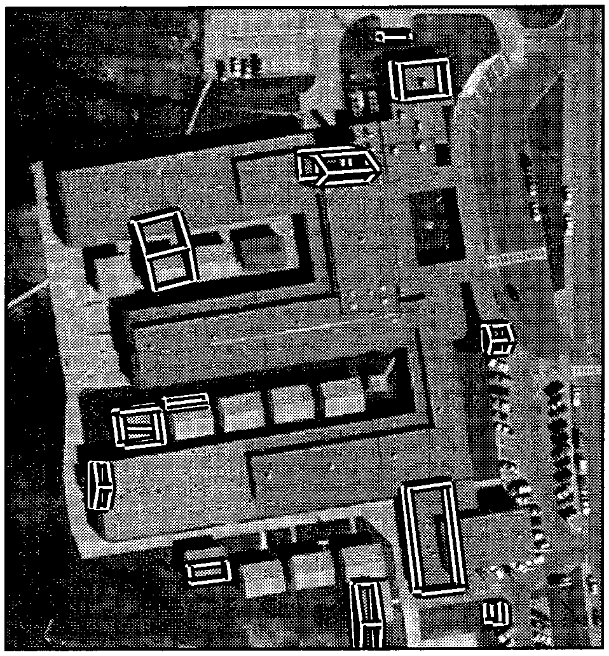


Figure B-151: VHBUILD results, NEST_FHN711

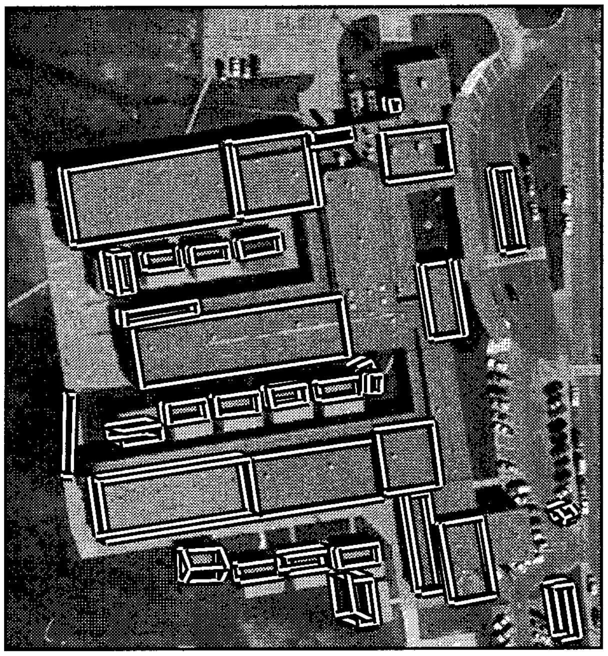


Figure B-152: PIVOT results, NEST_FHN711

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	56.96	**0.070**	**54.77**	-	-	-
BUILD+SHAVE	**61.35**	0.286	52.20	**43.65**	5.183	13.38
VHBUILD	12.07	0.258	11.71	6.99	1.381	6.37
PIVOT	57.49	0.148	52.99	34.76	**0.521**	**29.43**

Table B-42: 2D and 3D evaluation results for NEST_FHN711

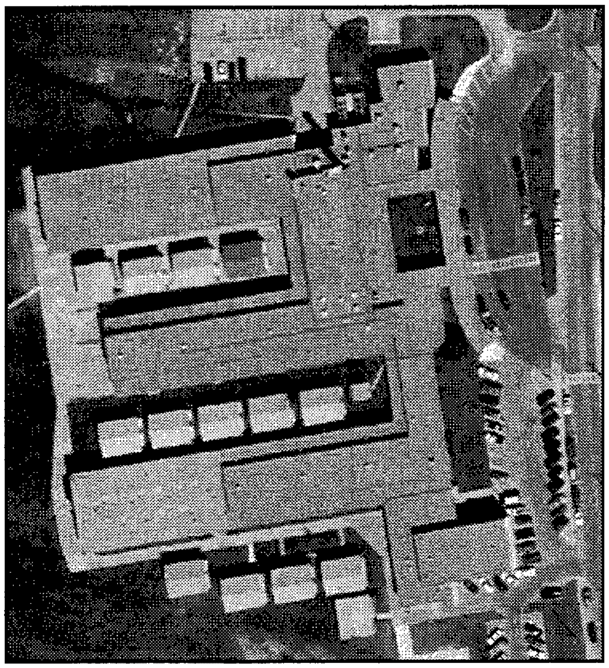


Figure B-153: NEST_FHN78 image

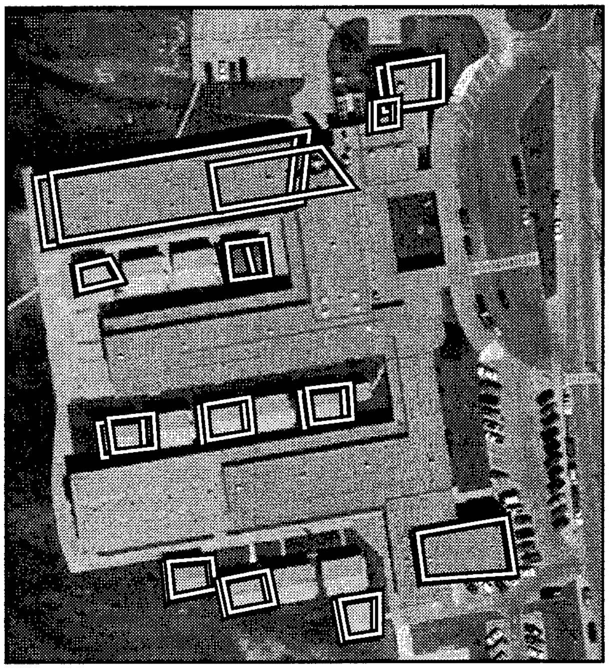


Figure B-154: BUILD+SHAVE results, NEST_FHN78

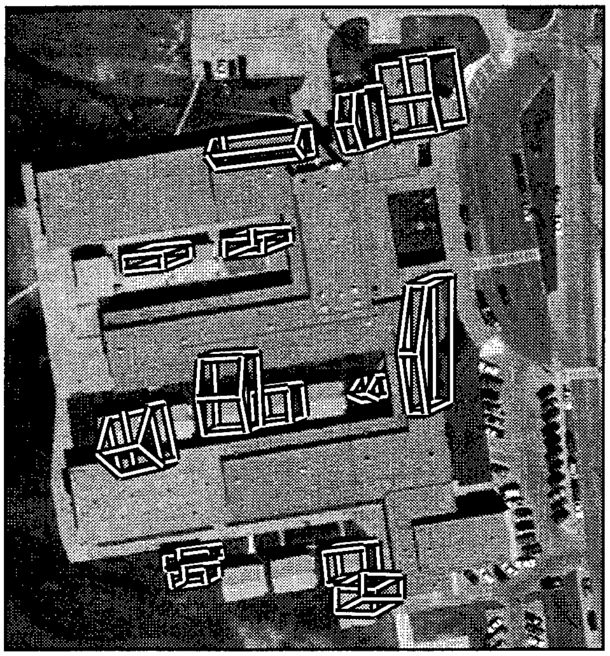


Figure B-155: VHBUILD results, NEST_FHN78

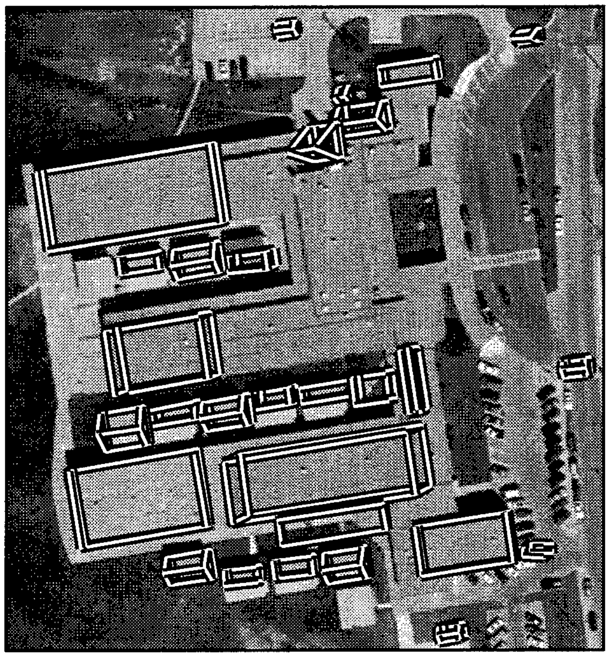


Figure B-156: PIVOT results, NEST_FHN78

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	27.16	**0.031**	26.93	-	-	-
BUILD+SHAVE	28.30	0.093	27.58	21.11	1.479	16.09
VHBUILD	19.77	0.427	18.23	5.85	11.647	3.48
PIVOT	**48.05**	0.155	**44.72**	**37.34**	**1.106**	**26.43**

Table B-43: 2D and 3D evaluation results for NEST_FHN78

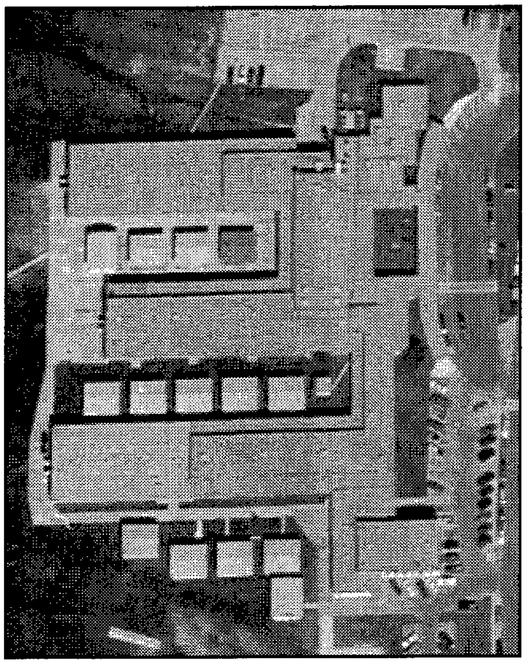


Figure B-165: NEST_FHOV625 image

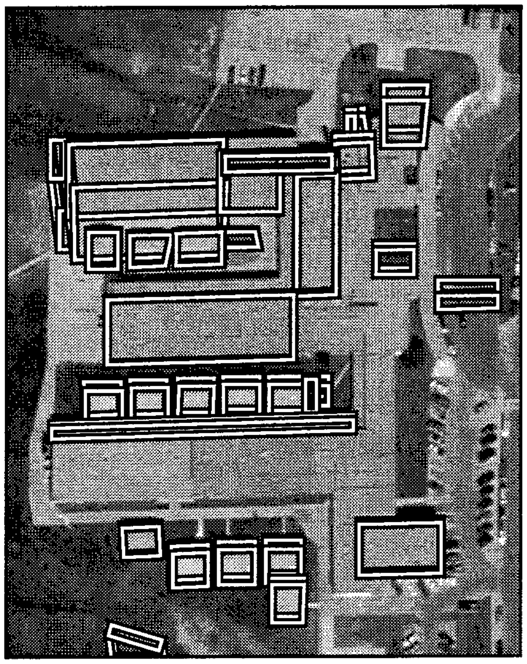


Figure B-166: BUILD+SHAVE results, NEST_FHOV625

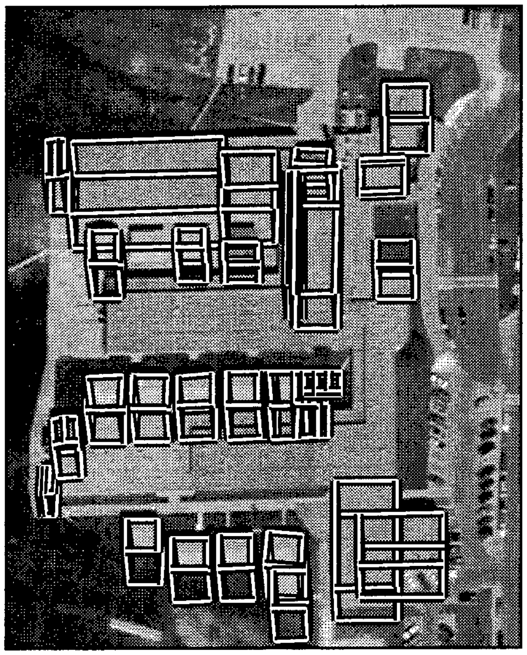


Figure B-167: VHBUILD results, NEST_FHOV625

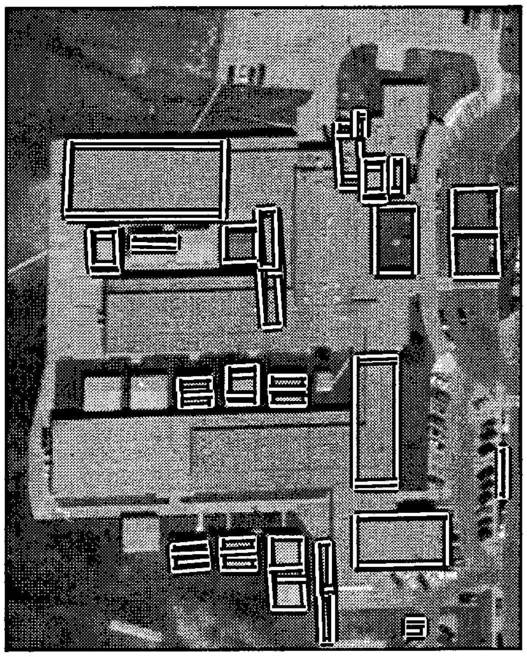


Figure B-168: PIVOT results, NEST_FHOV625

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	49.42	**0.082**	**47.49**	-	-	-
BUILD+SHAVE	52.76	0.210	**47.49**	24.97	4.903	11.23
VHBUILD	**54.55**	0.325	46.33	**25.41**	11.618	6.43
PIVOT	31.57	0.355	28.39	21.34	**3.124**	**12.80**

Table B-46: 2D and 3D evaluation results for NEST_FHOV625

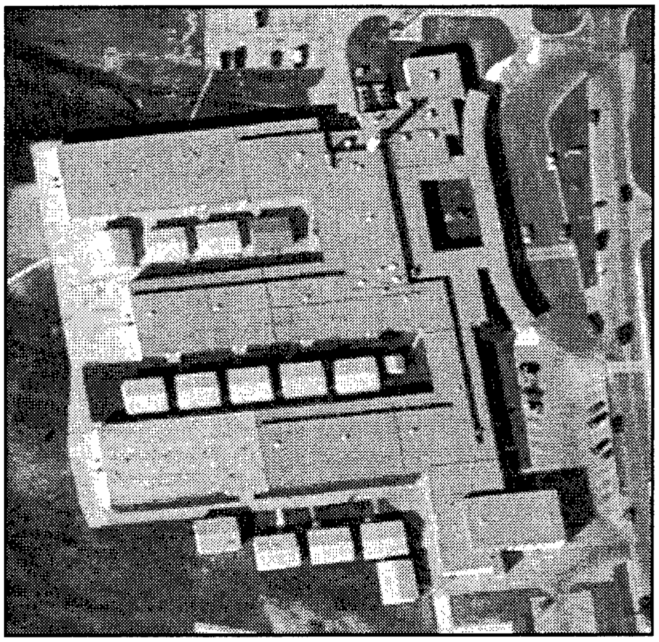


Figure B-169: NEST_FHOV927 image

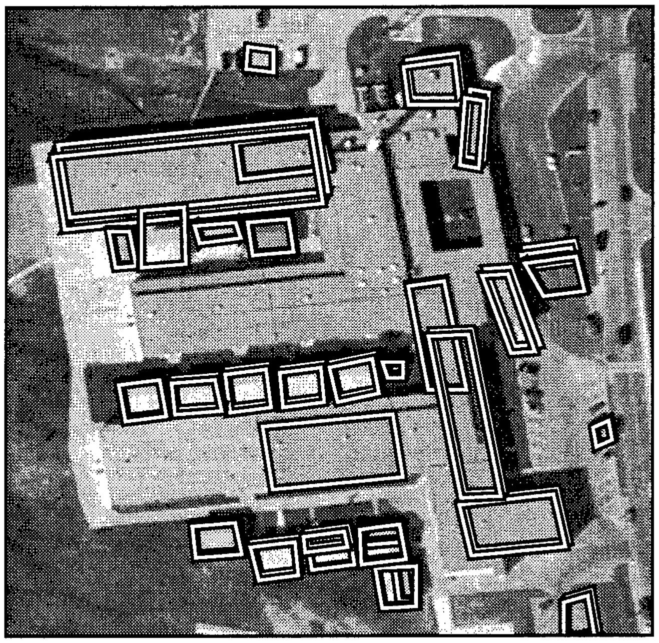


Figure B-170: BUILD+SHAVE results, NEST_FHOV927

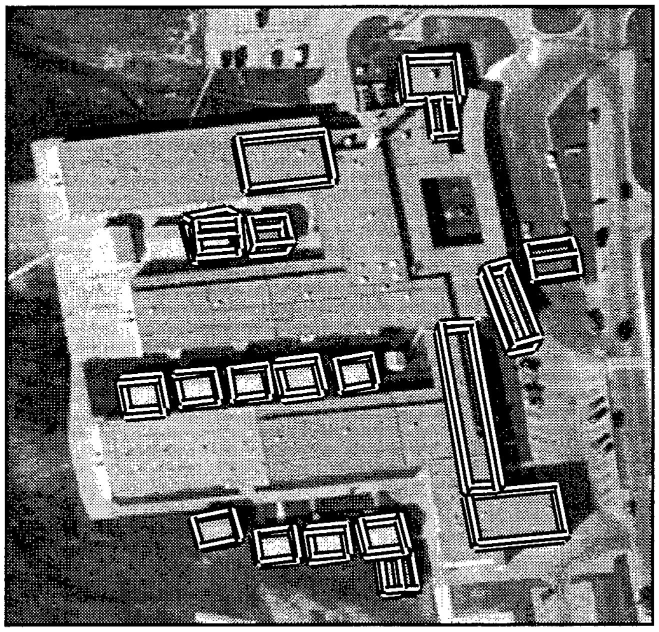


Figure B-171: VHBUILD results, NEST_FHOV927

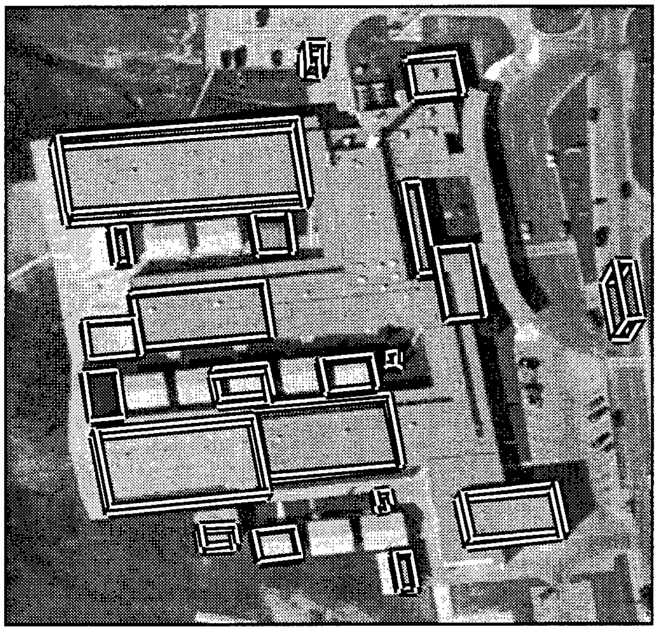


Figure B-172: PIVOT results, NEST_FHOV927

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	45.54	**0.118**	43.21	-	-	-
BUILD+SHAVE	49.02	0.172	45.21	25.27	2.128	16.43
VHBUILD	27.33	0.196	25.93	20.25	**1.177**	16.35
PIVOT	**49.64**	0.196	**45.24**	**40.06**	1.253	**26.67**

Table B-47: 2D and 3D evaluation results for NEST_FHOV927

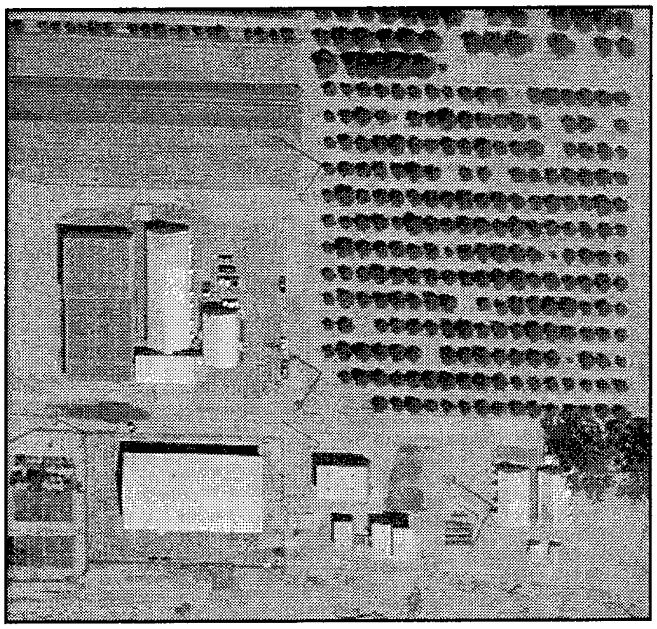


Figure B-173: ORCHARD_RI0607 image

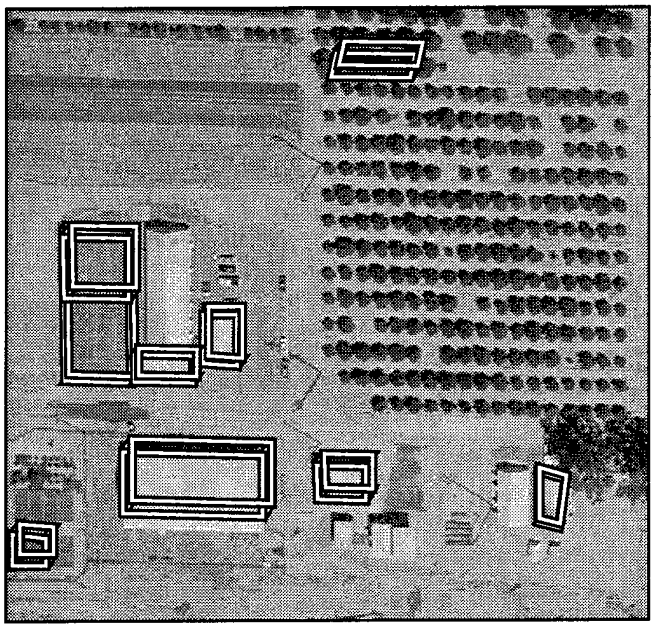


Figure B-174: BUILD+SHAVE results, ORCHARD_RI0607

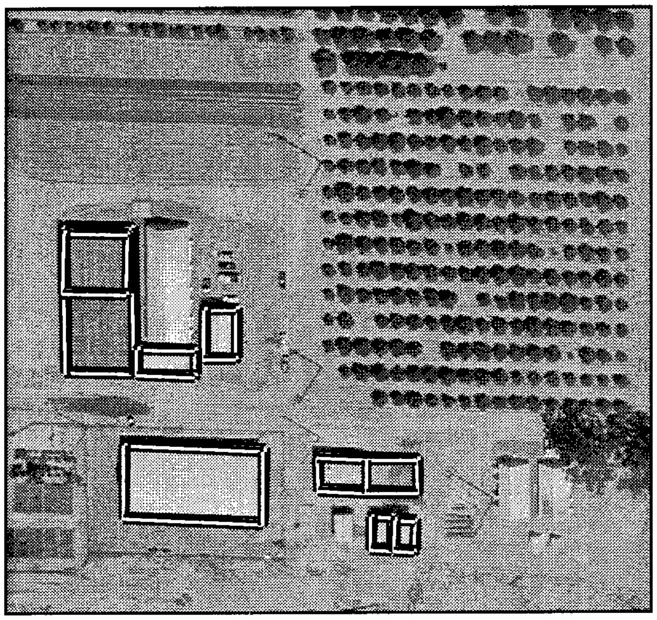


Figure B-175: VHBUILD results, ORCHARD_RI0607

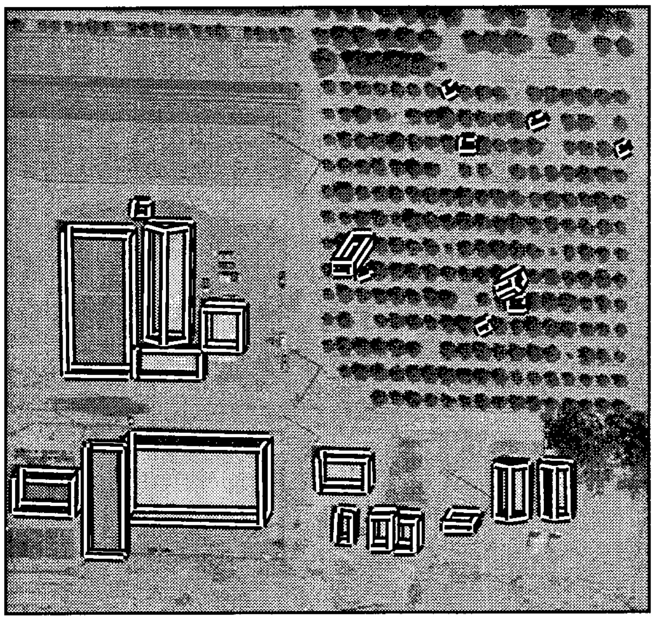


Figure B-176: PIVOT results, ORCHARD_RI0607

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	63.94	**0.125**	59.19	-	-	-
BUILD+SHAVE	68.77	0.319	56.40	69.35	1.107	39.23
VHBUILD	71.21	0.133	65.06	38.73	**0.293**	34.77
PIVOT	**96.07**	0.438	**67.60**	**93.33**	1.110	**45.84**

Table B-48: 2D and 3D evaluation results for ORCHARD_RI0607

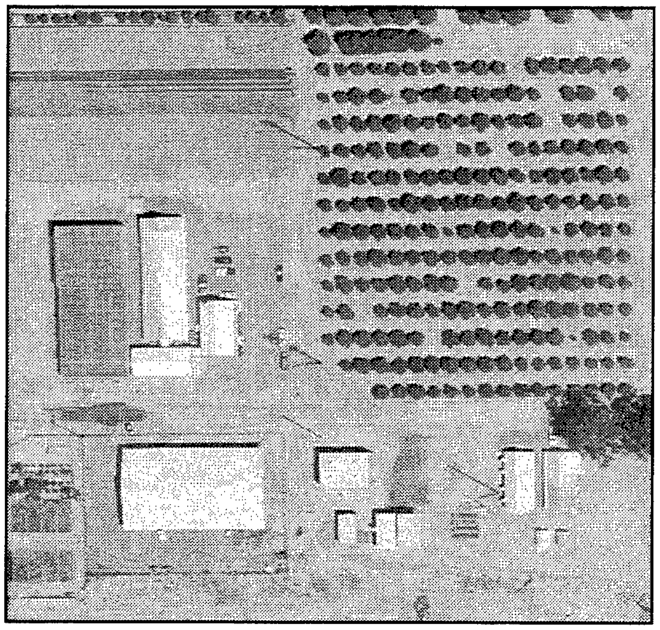


Figure B-177: ORCHARD_RI0608 image

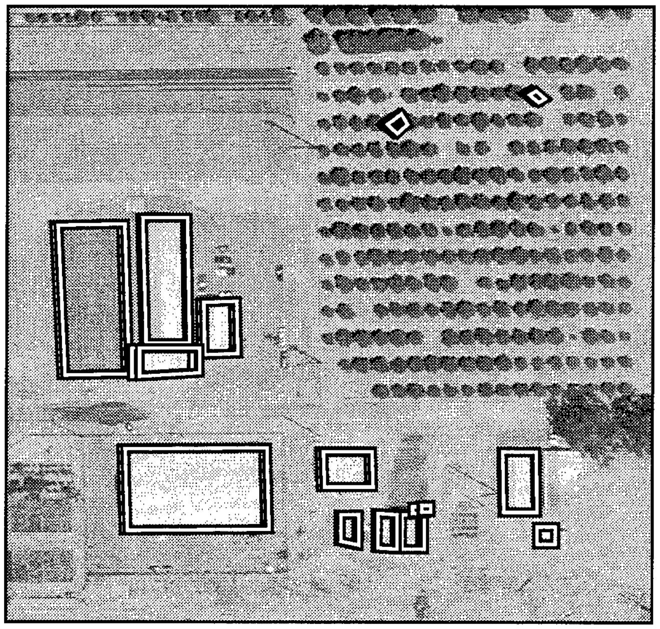


Figure B-178: BUILD+SHAVE results, ORCHARD_RI0608

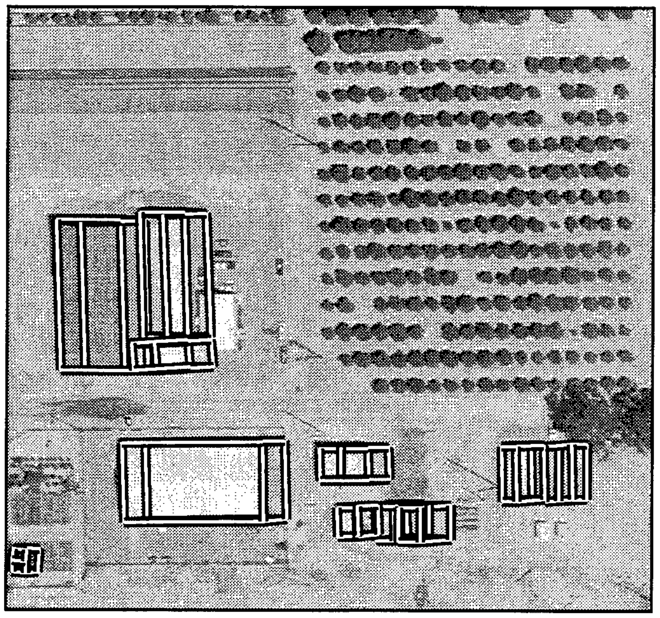


Figure B-179: VHBUILD results, ORCHARD_RI0608

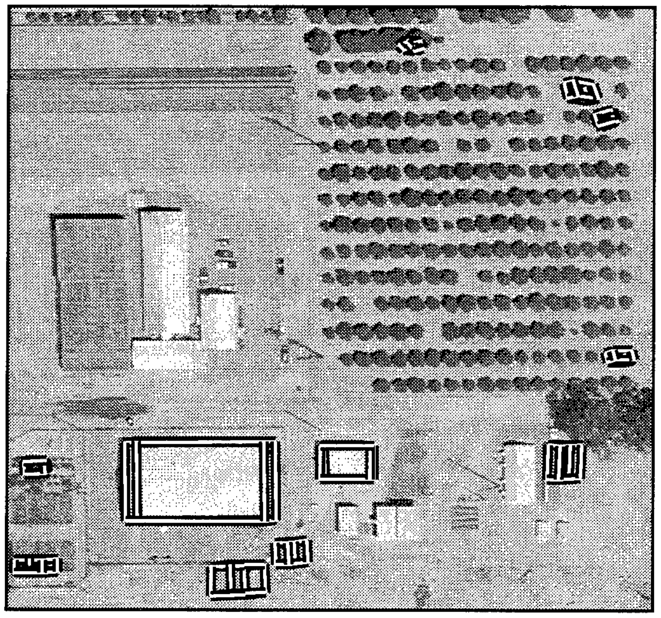


Figure B-180: PIVOT results, ORCHARD_RI0608

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	88.83	**0.059**	84.39	-	-	-
BUILD+SHAVE	92.11	0.087	**85.28**	**68.69**	**0.352**	**55.30**
VHBUILD	**94.33**	0.293	73.93	67.36	7.263	11.43
PIVOT	35.67	0.433	30.90	35.07	2.822	17.63

Table B-49: 2D and 3D evaluation results for ORCHARD_RI0608

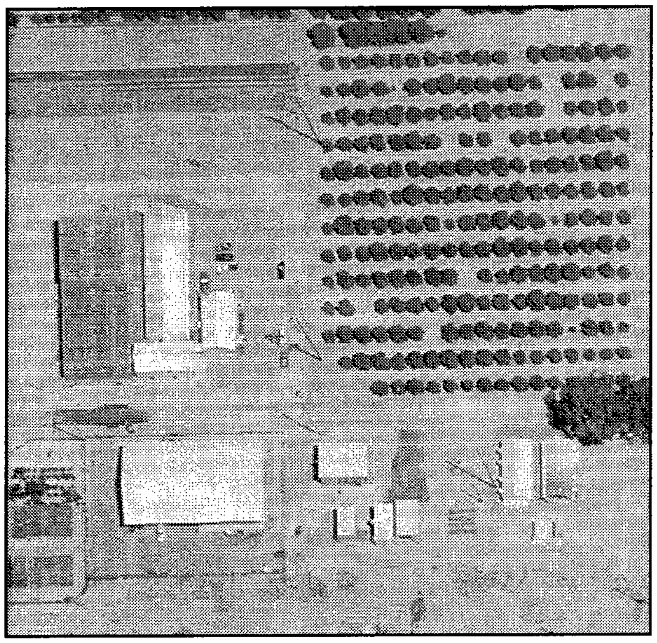


Figure B-181: ORCHARD_RI0609 image

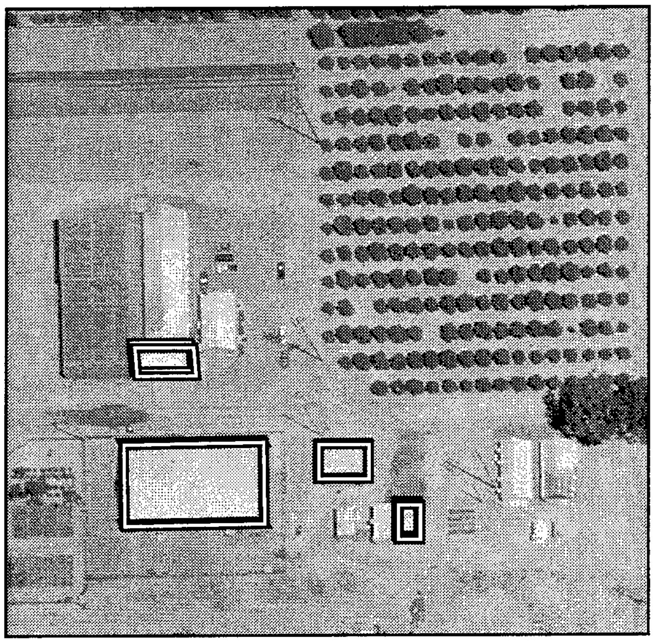


Figure B-182: BUILD+SHAVE results, ORCHARD_RI0609

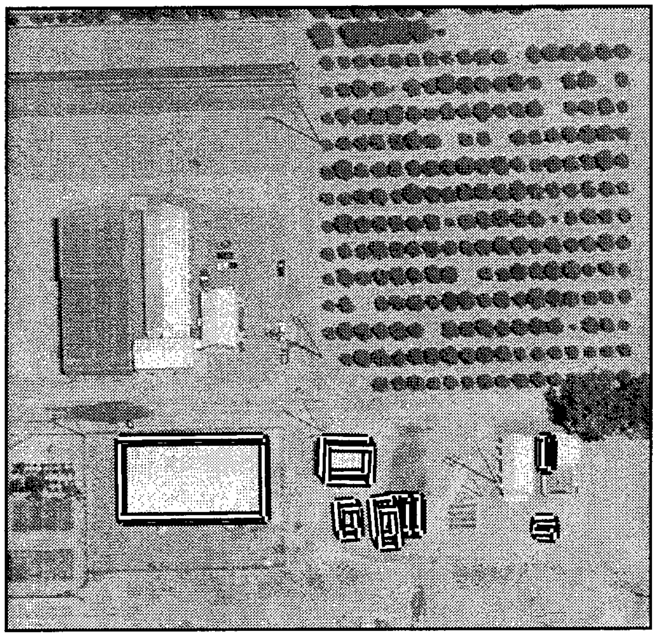


Figure B-183: VHBUILD results, ORCHARD_RI0609

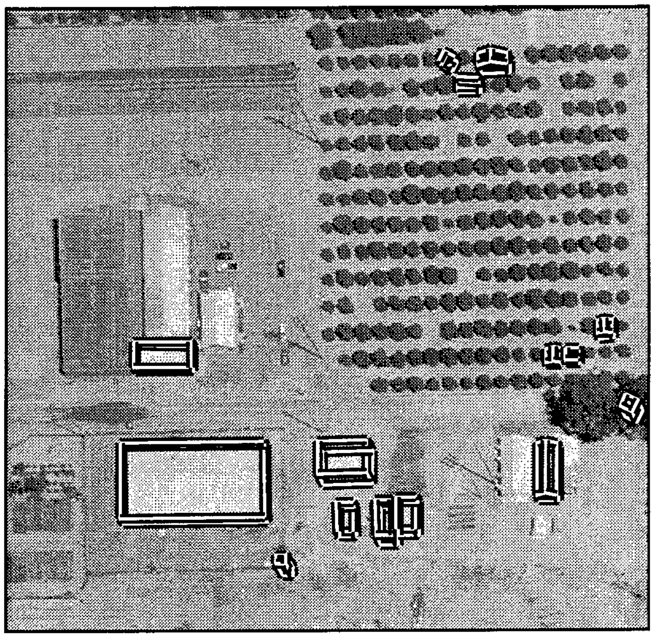


Figure B-184: PIVOT results, ORCHARD_RI0609

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	35.57	**0.049**	34.96	-	-	-
BUILD+SHAVE	36.83	0.084	35.72	7.14	1.084	6.63
VHBUILD	37.28	0.167	35.09	13.31	1.075	11.65
PIVOT	**41.97**	0.309	**37.15**	**22.40**	**0.916**	**18.59**

Table B-50: 2D and 3D evaluation results for ORCHARD_RI0609

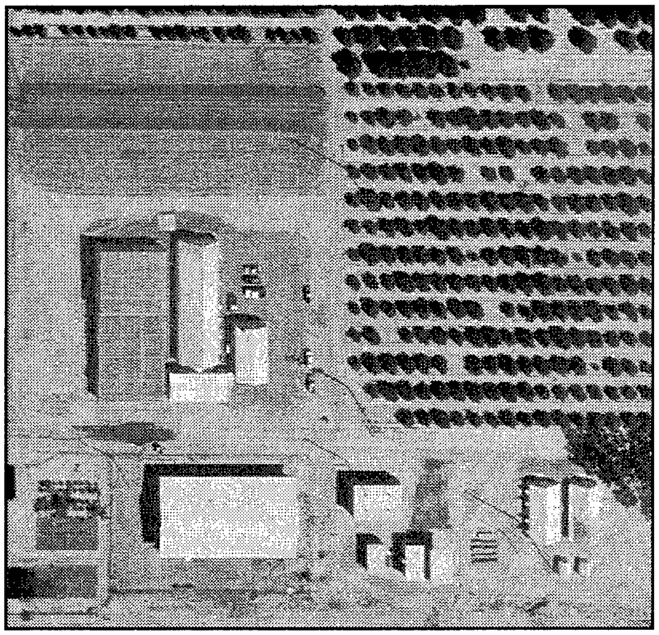


Figure B-185: ORCHARD_RI0707 image

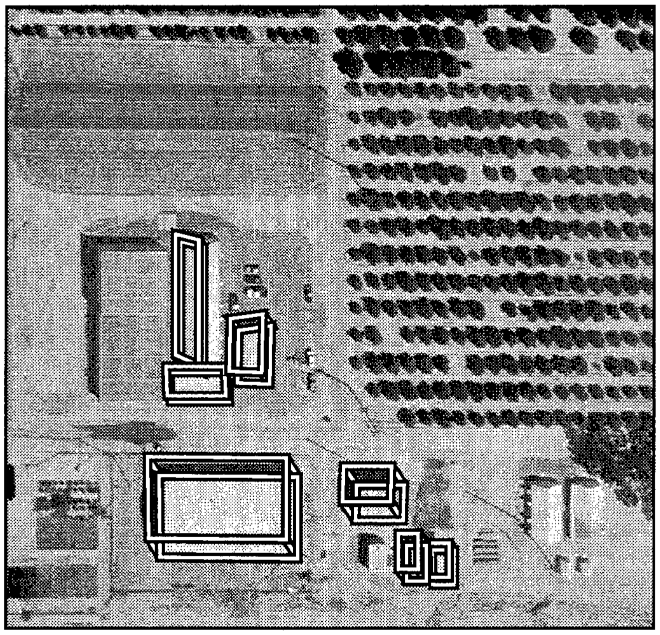


Figure B-186: BUILD+SHAVE results, ORCHARD_RI0707

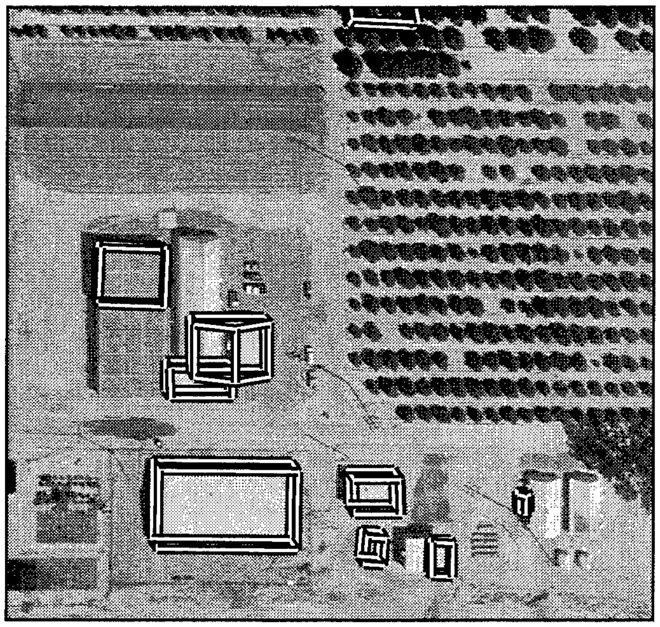


Figure B-187: VHBUILD results, ORCHARD_RI0707

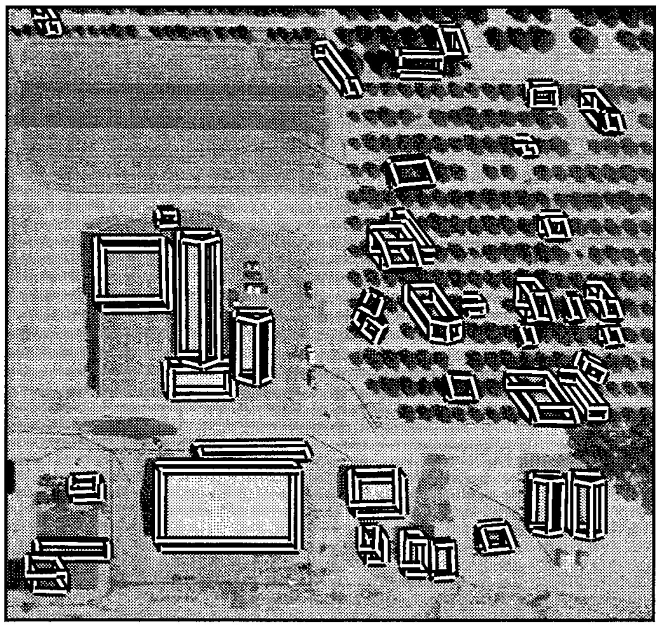


Figure B-188: PIVOT results, ORCHARD_RI0707

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	49.56	**0.045**	48.47	-	-	-
BUILD+SHAVE	55.54	0.273	48.23	49.08	1.449	28.68
VHBUILD	59.65	0.251	**51.89**	51.86	**0.707**	**37.95**
PIVOT	**81.68**	0.947	46.05	**73.10**	1.389	36.28

Table B-51: 2D and 3D evaluation results for ORCHARD_RI0707

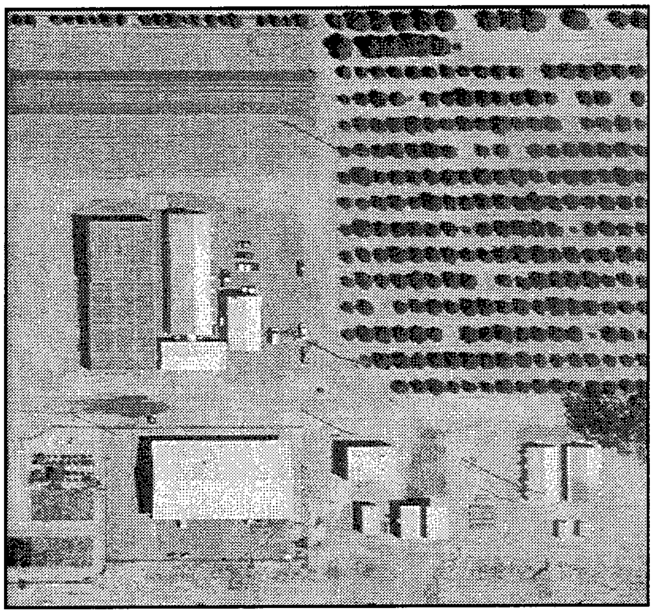


Figure B-189: ORCHARD_RI0708 image

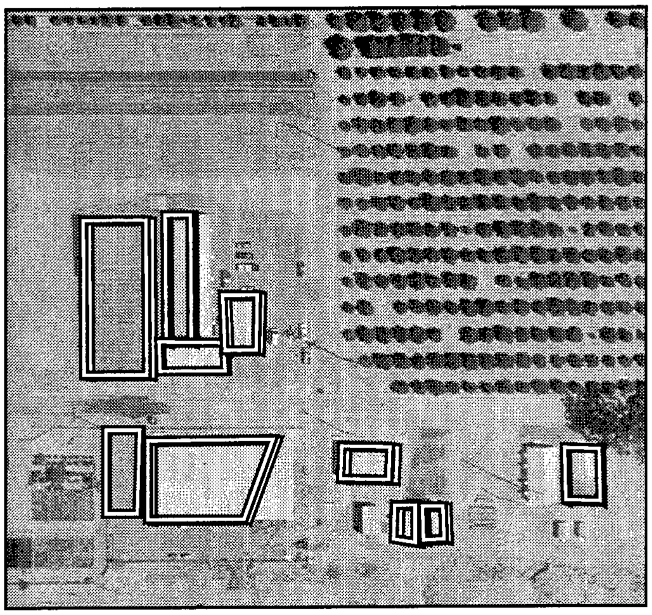


Figure B-190: BUILD+SHAVE results, ORCHARD_RI0708

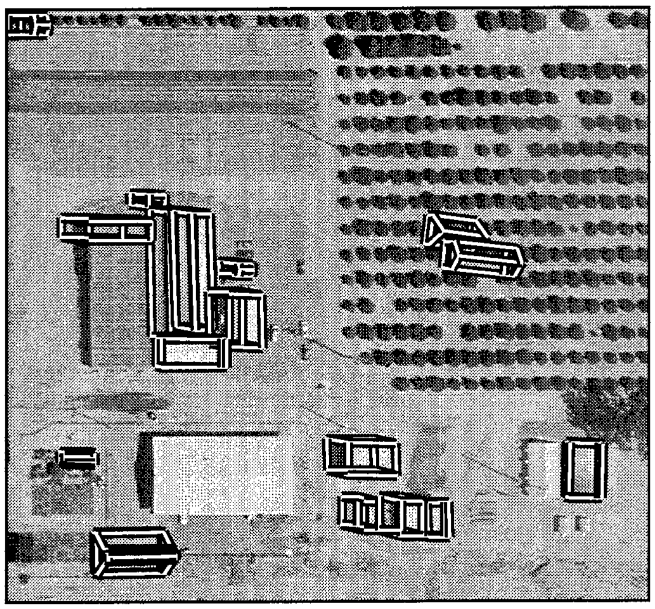


Figure B-191: VHBUILD results, ORCHARD_RI0708

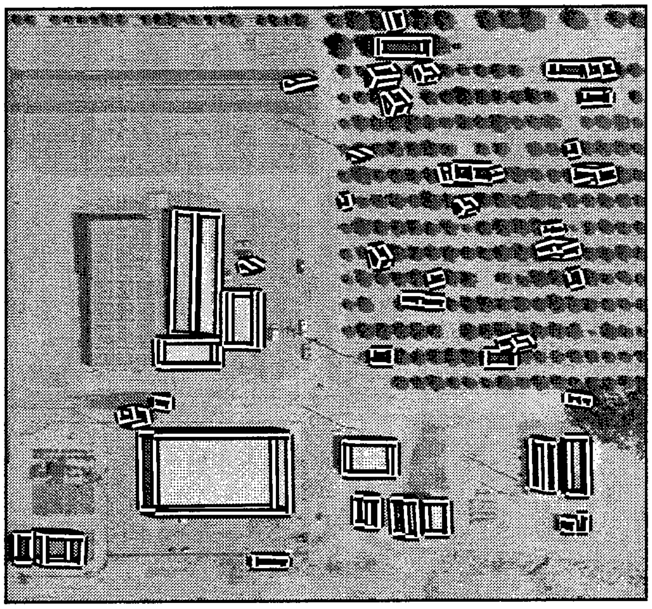


Figure B-192: PIVOT results, ORCHARD_RI0708

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	76.61	**0.159**	**68.28**	-	-	-
BUILD+SHAVE	**79.44**	0.224	67.46	**70.33**	**0.848**	**44.05**
VHBUILD	43.37	0.930	30.91	26.15	9.628	7.43
PIVOT	65.43	0.756	43.78	56.78	4.154	16.91

Table B-52: 2D and 3D evaluation results for ORCHARD_RI0708

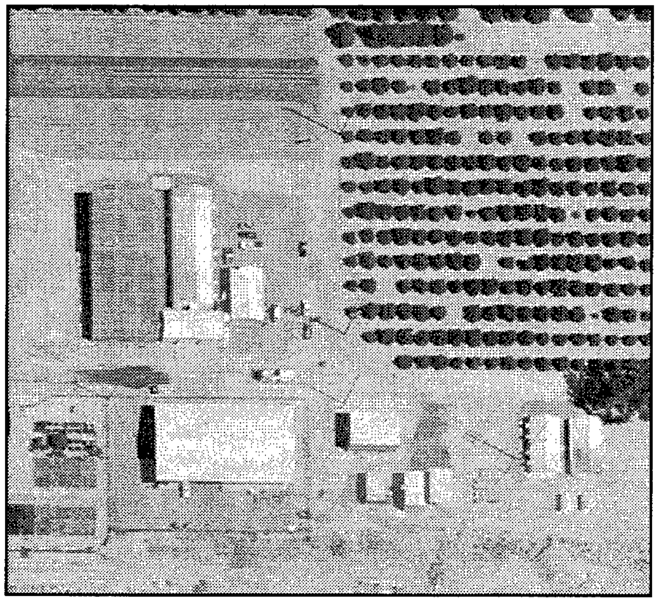


Figure B-193: ORCHARD_RI0709 image

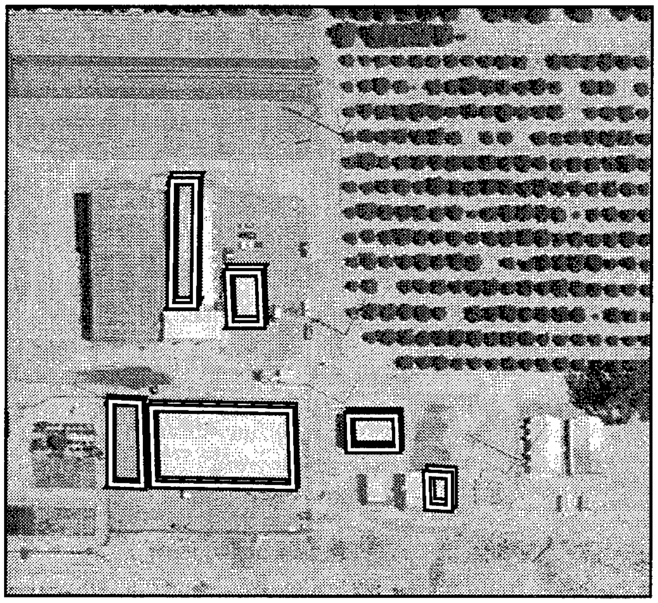


Figure B-194: BUILD+SHAVE results, ORCHARD_RI0709

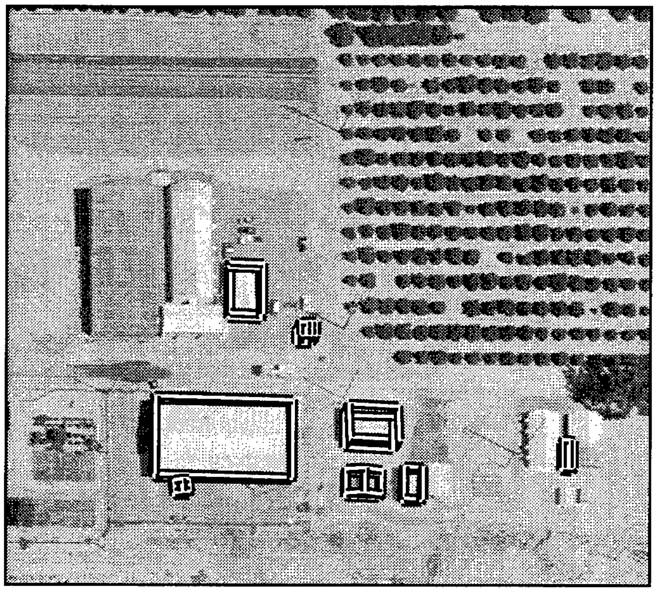


Figure B-195: VHBUILD results, ORCHARD_RI0709

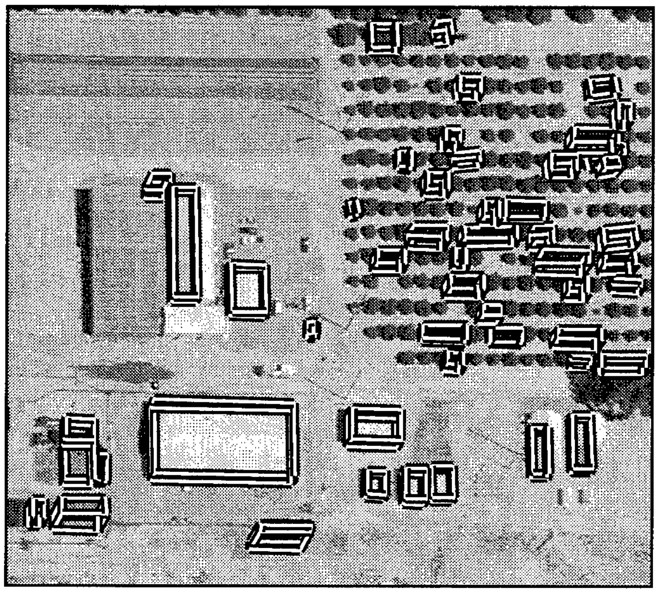


Figure B-196: PIVOT results, ORCHARD_RI0709

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.75	0.216	39.14	-	-	-
BUILD+SHAVE	45.42	0.267	**40.51**	27.08	0.630	23.13
VHBUILD	41.95	**0.141**	39.61	22.35	**0.583**	19.77
PIVOT	**54.18**	1.439	30.44	**46.16**	1.692	**25.92**

Table B-53: 2D and 3D evaluation results for ORCHARD_RI0709

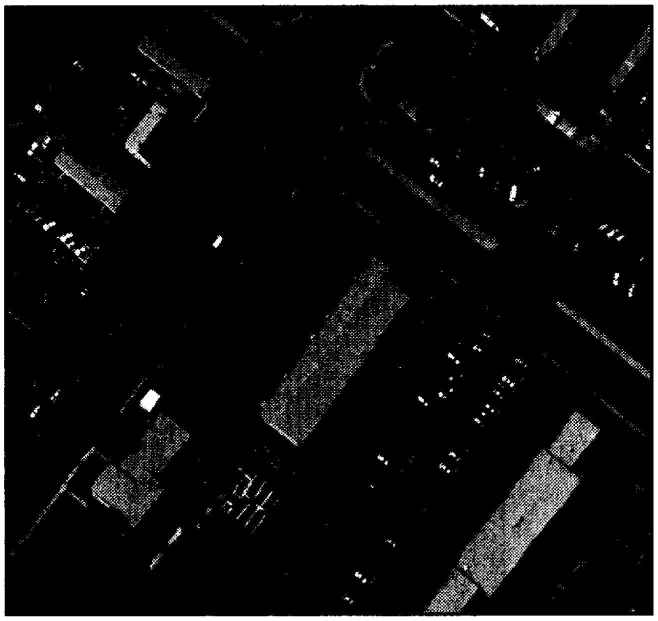


Figure B-197: RADT10 image

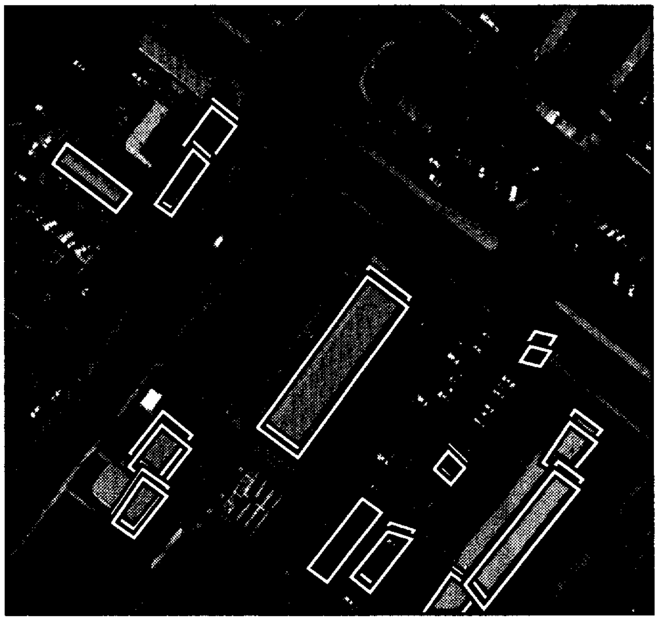


Figure B-198: BUILD+SHAVE results, RADT10

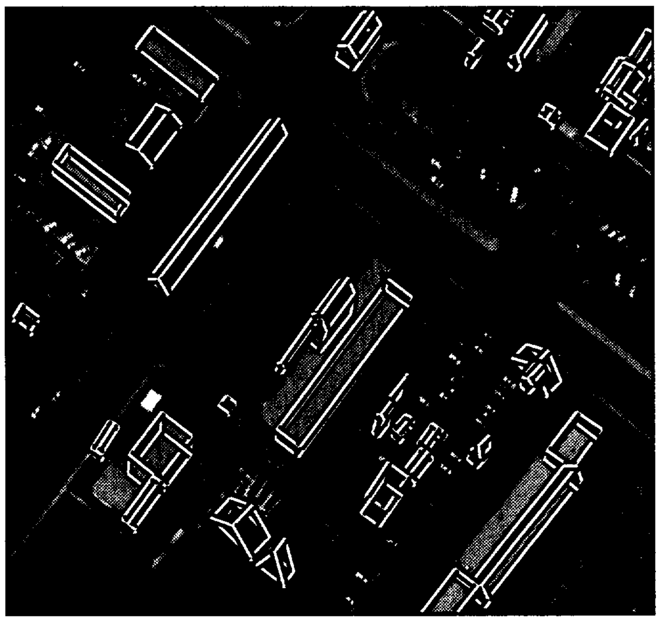


Figure B-199; VHBUILD results, RADT10

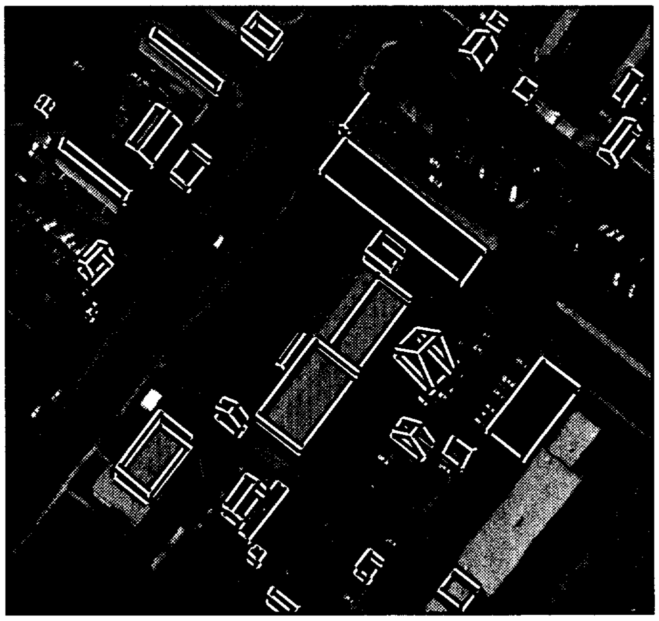


Figure B-200: PIVOT results, RADT10

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	51.47	**0.343**	43.74	-	-	-
BUILD+SHAVE	55.96	0.450	**44.71**	**52.94**	**1.147**	**32.93**
VHBUILD	**65.29**	0.798	42.93	51.74	2.280	23.74
PIVOT	48.38	1.396	28.88	34.85	2.085	20.18

Table B-54: 2D and 3D evaluation results for RADT10

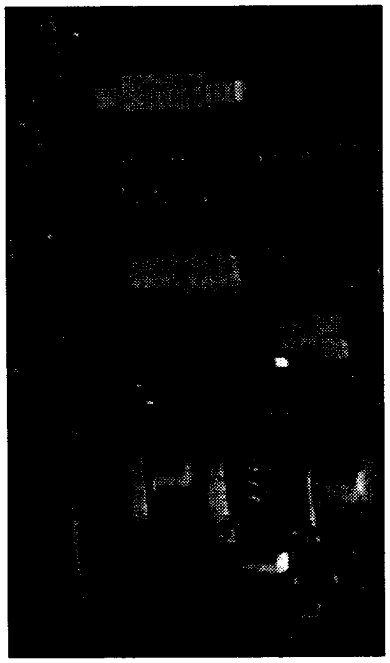


Figure B-201: RADT10OB image

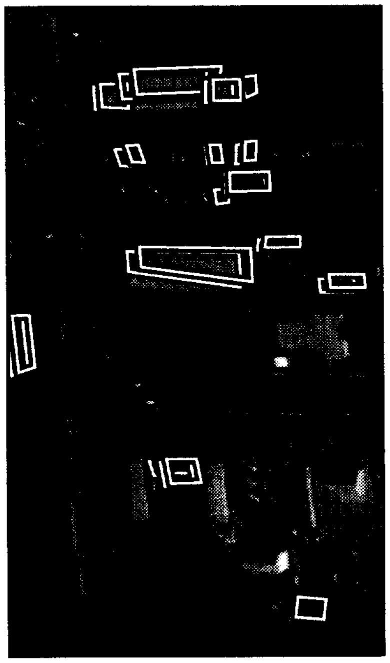


Figure B-202: BUILD+SHAVE results, RADT10OB

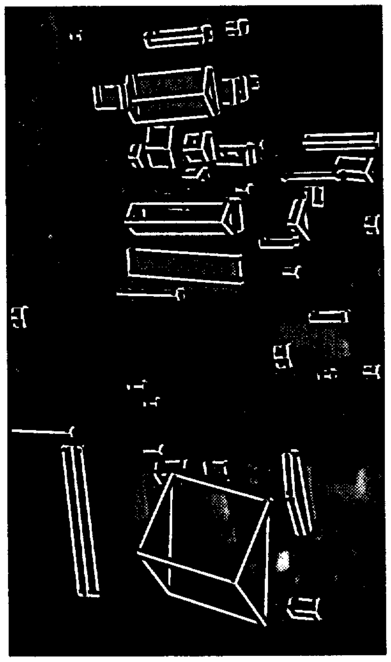


Figure B-203: VHBUILD results, RADT10OB

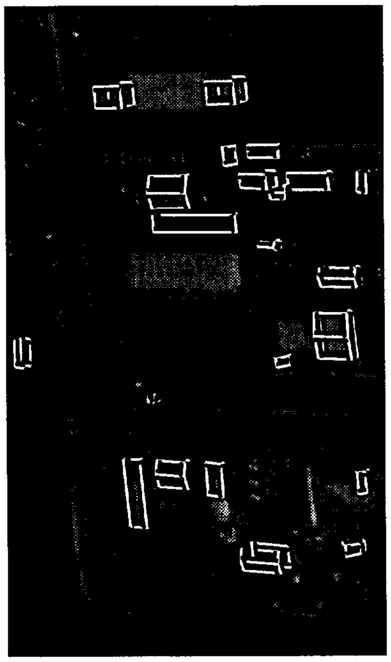


Figure B-204: PIVOT results, RADT10OB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	38.51	**0.390**	33.48	-	-	-
BUILD+SHAVE	44.60	0.617	**34.98**	41.34	**0.926**	**29.89**
VHBUILD	**60.13**	2.600	23.45	**42.64**	5.817	12.25
PIVOT	35.27	1.183	24.89	25.01	1.047	19.82

Table B-55: 2D and 3D evaluation results for RADT10OB

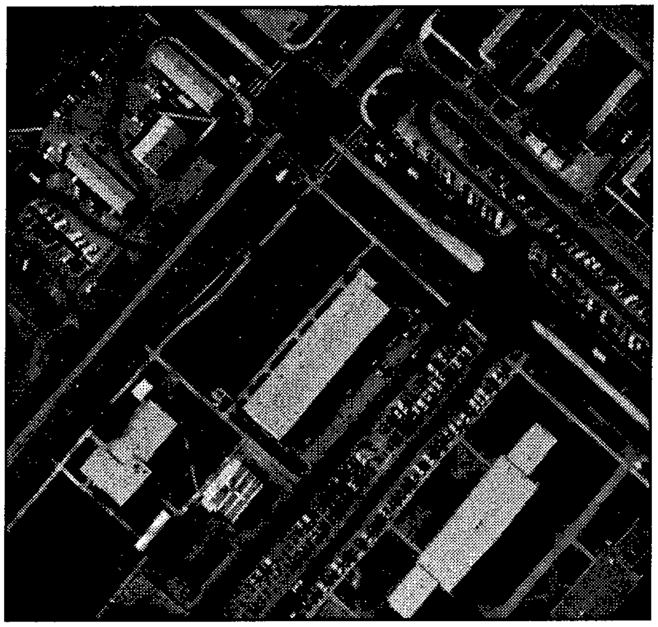


Figure B-205: RADT10S image

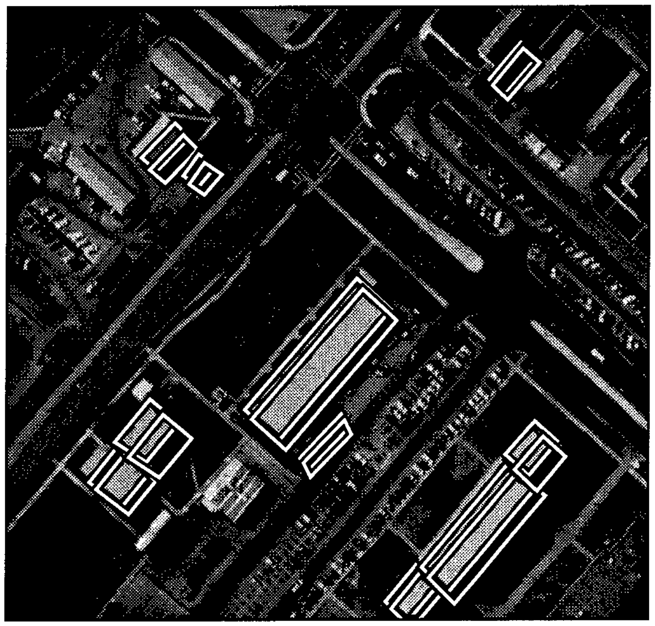


Figure B-206: BUILD+SHAVE results, RADT10S

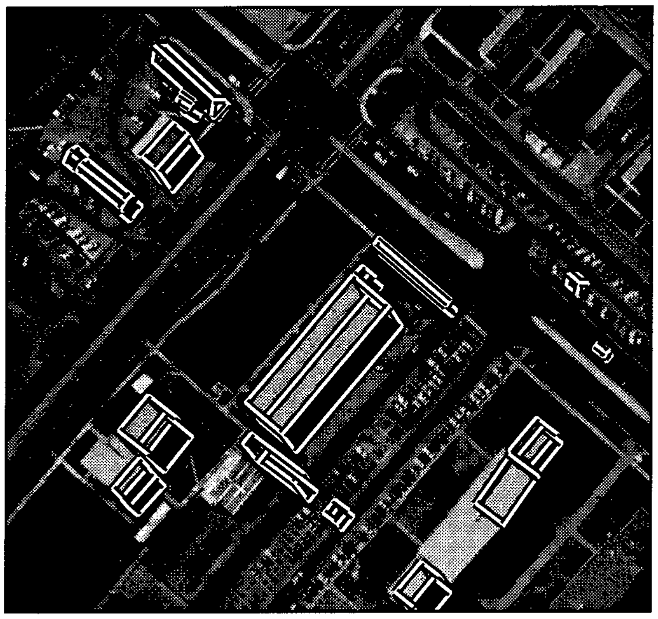


Figure B-207: VHBUILD results, RADT10S

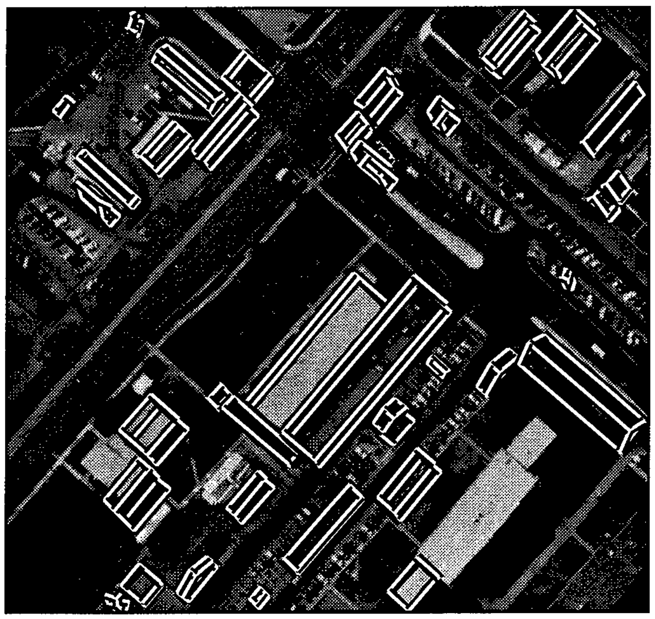


Figure B-208: PIVOT results, RADT10S

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	47.16	**0.137**	44.29	-	-	-
BUILD+SHAVE	**56.58**	0.281	**48.82**	**53.60**	**1.166**	**32.99**
VHBUILD	51.95	0.506	41.14	30.02	4.281	13.14
PIVOT	51.87	1.820	26.68	34.36	3.820	14.86

Table B-56: 2D and 3D evaluation results for RADT10S

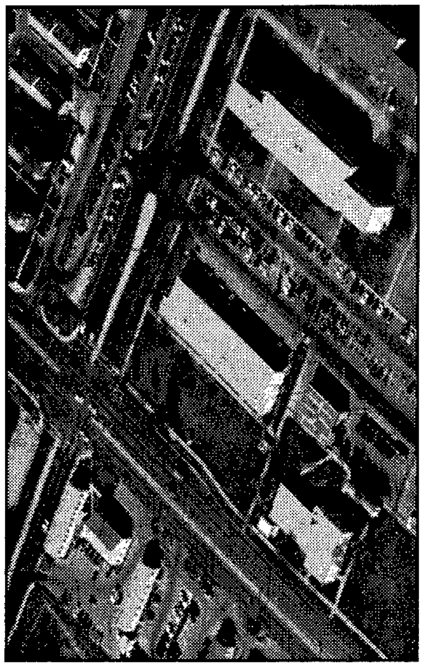


Figure B-209: RADT10WOB image

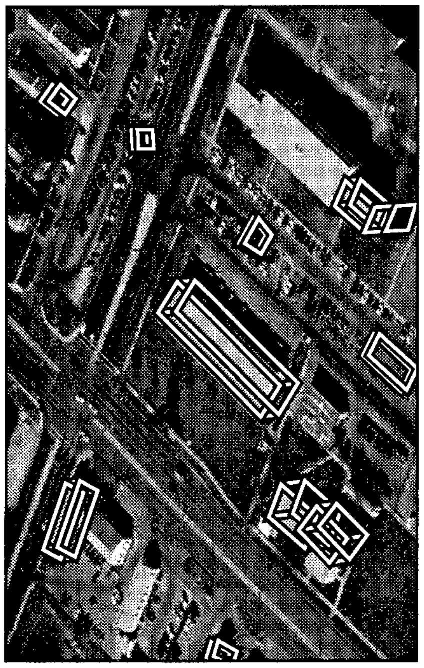


Figure B-210: BUILD+SHAVE results, RADT10WOB

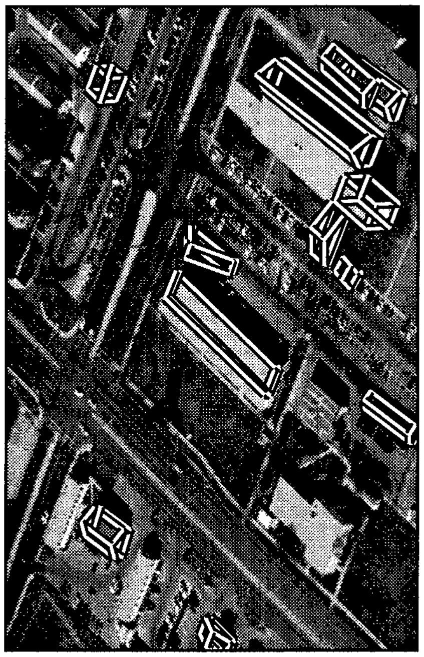


Figure B-211: VHBUILD results, RADT10WOB

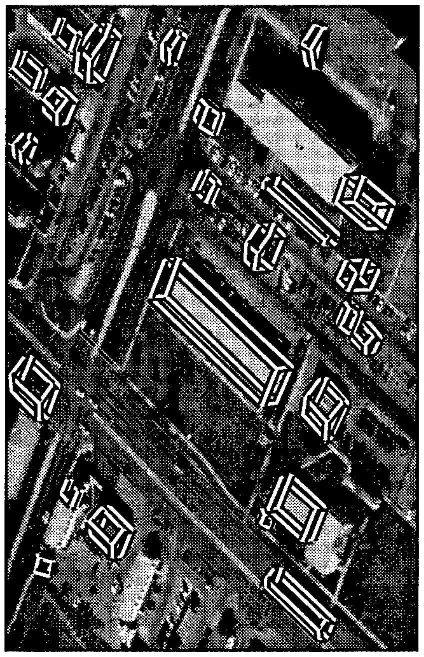


Figure B-212: PIVOT results, RADT10WOB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	31.47	**0.191**	29.68	-	-	-
BUILD+SHAVE	42.06	0.369	**36.40**	29.29	1.114	22.09
VHBUILD	28.67	1.224	21.22	16.03	2.336	11.66
PIVOT	**46.79**	0.892	33.01	**35.11**	**1.009**	**25.93**

Table B-57: 2D and 3D evaluation results for RADT10WOB

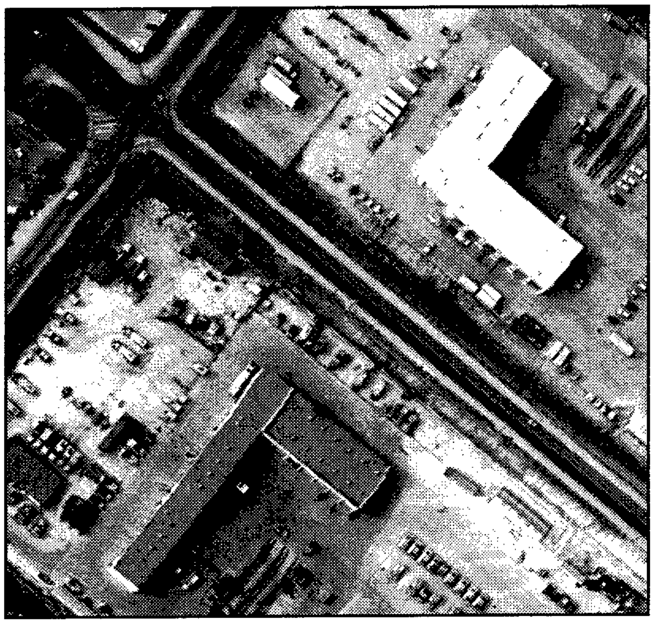


Figure B-213: RADT11 image

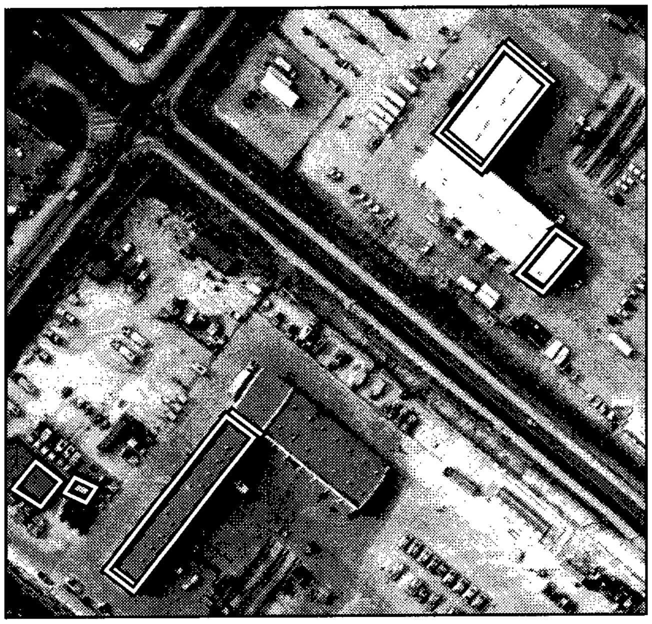


Figure B-214: BUILD+SHAVE results, RADT11

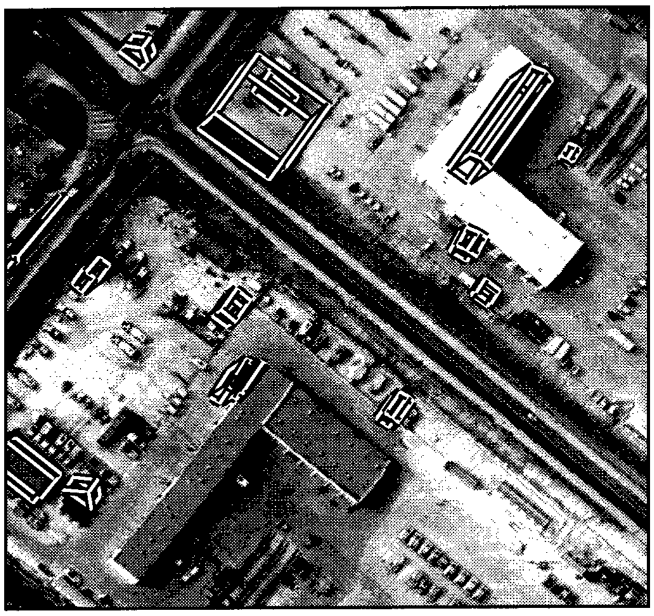


Figure B-215: VHBUILD results, RADT11

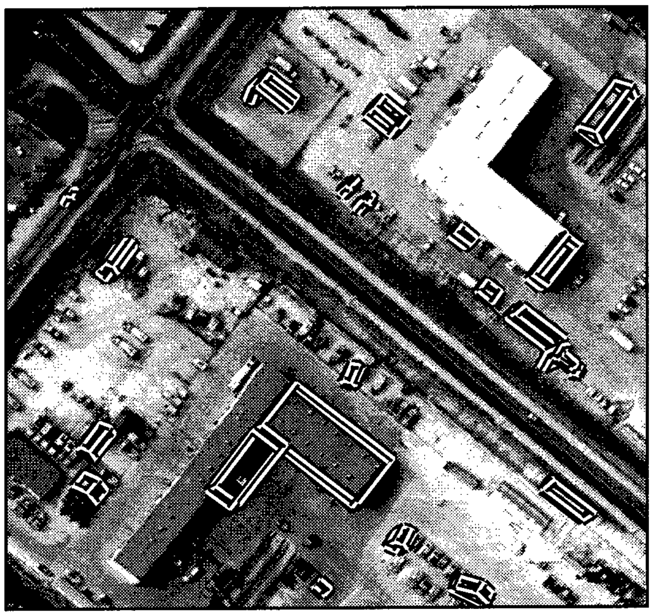


Figure B-216: PIVOT results, RADT11

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.41	**0.094**	40.79	-	-	-
BUILD+SHAVE	**45.59**	0.113	**43.37**	**33.53**	**0.204**	**31.38**
VHBUILD	17.76	2.560	12.21	10.75	10.669	5.01
PIVOT	25.49	1.616	18.05	21.57	1.967	15.14

Table B-58: 2D and 3D evaluation results for RADT11

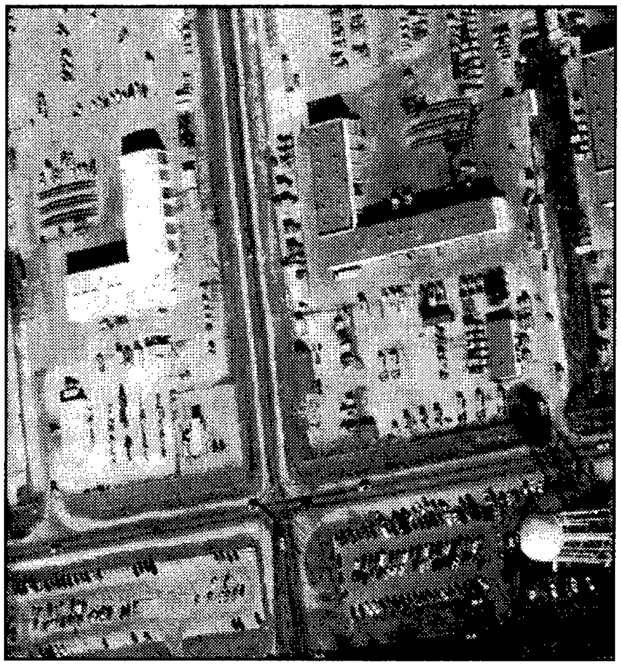


Figure B-217: RADT11OB image

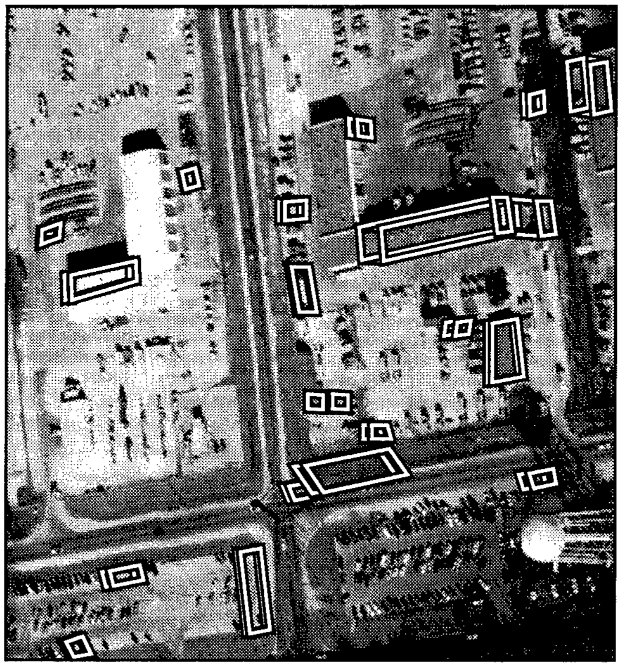


Figure B-218: BUILD+SHAVE results, RADT11OB

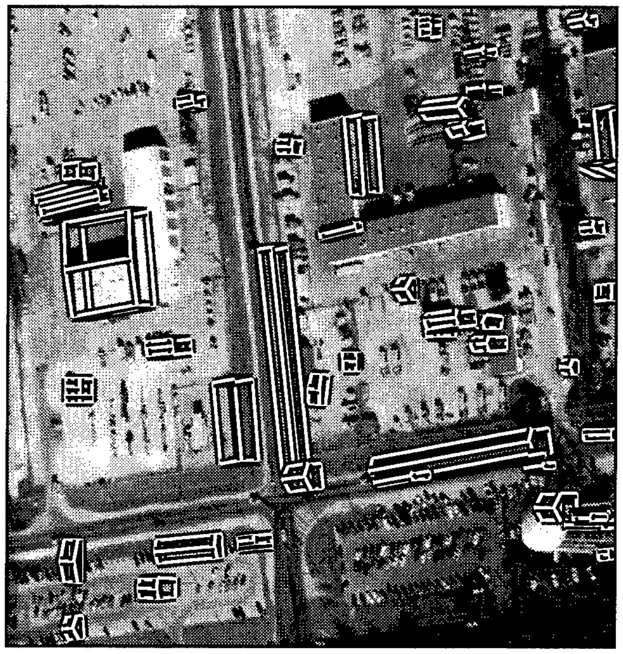


Figure B-219: VHBUILD results, RADT11OB

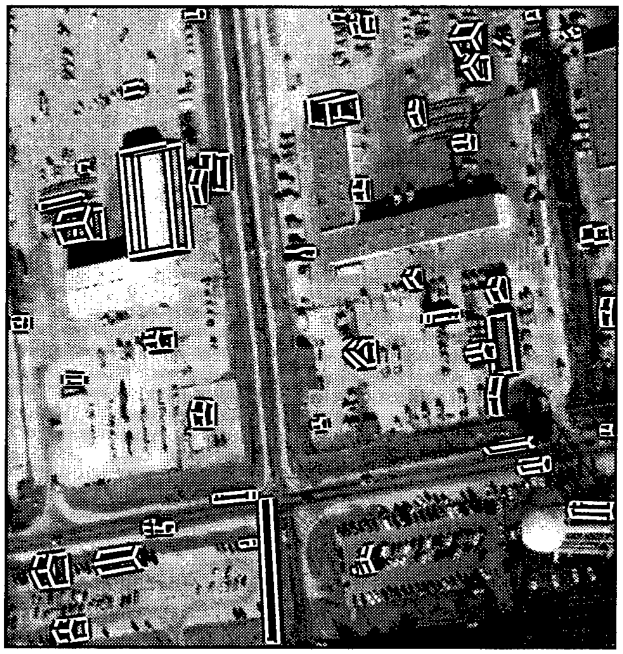


Figure B-220: PIVOT results, RADT11OB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	31.51	**1.347**	**22.12**	-	-	-
BUILD+SHAVE	**35.02**	1.894	21.06	23.30	2.962	13.79
VHBUILD	29.16	5.658	11.00	19.08	9.500	6.78
PIVOT	29.58	3.034	15.59	**25.25**	**2.317**	**15.93**

Table B-59: 2D and 3D evaluation results for RADT11OB

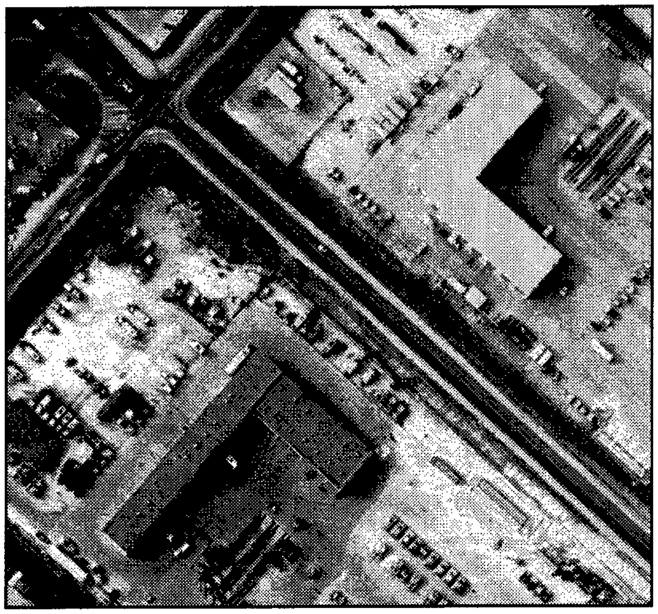


Figure B-221: RADT11S image

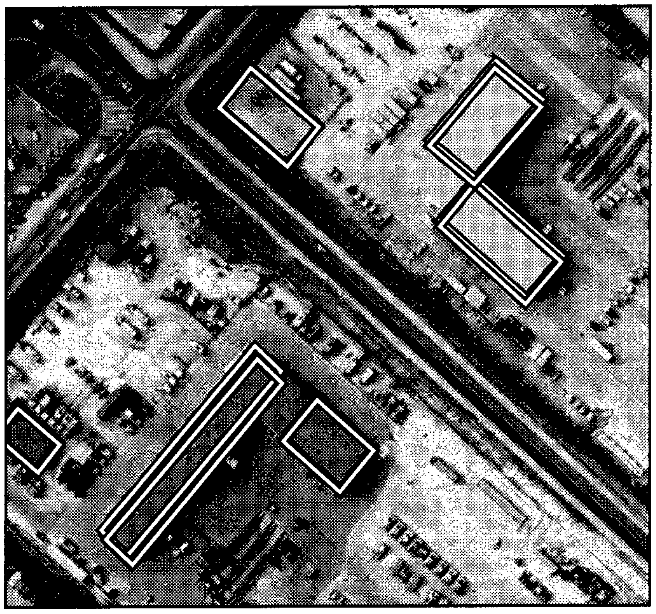


Figure B-222: BUILD+SHAVE results, RADT11S

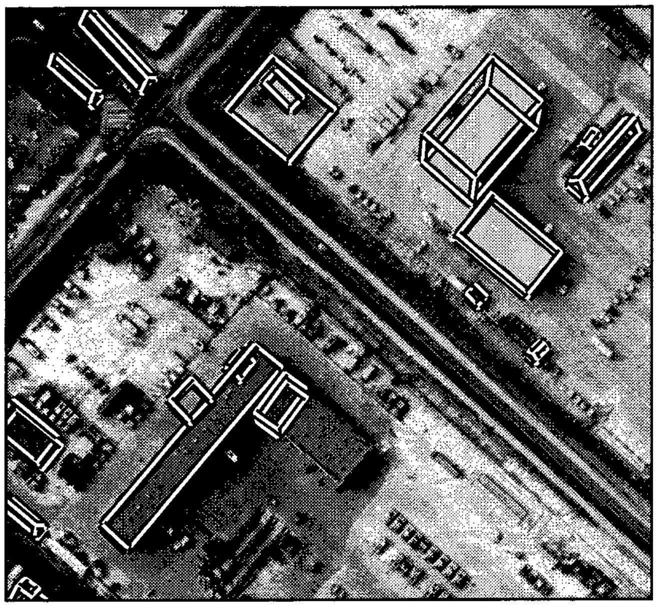


Figure B-223: VHBUILD results, RADT11S

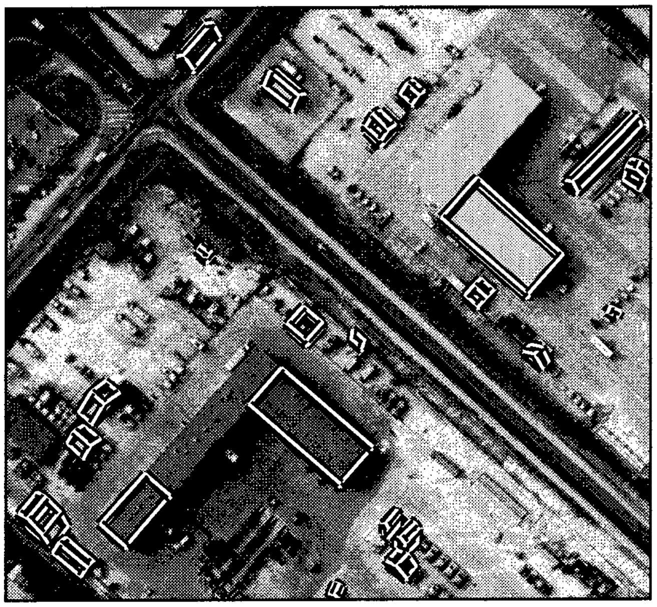


Figure B-224: PIVOT results, RADT11S

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	80.88	**0.274**	**66.19**	-	-	-
BUILD+SHAVE	**85.51**	0.359	65.43	**75.68**	1.444	**36.16**
VHBUILD	77.52	0.799	47.87	53.72	3.832	17.56
PIVOT	48.73	0.906	33.80	51.27	**1.280**	30.95

Table B-60: 2D and 3D evaluation results for RADT11S

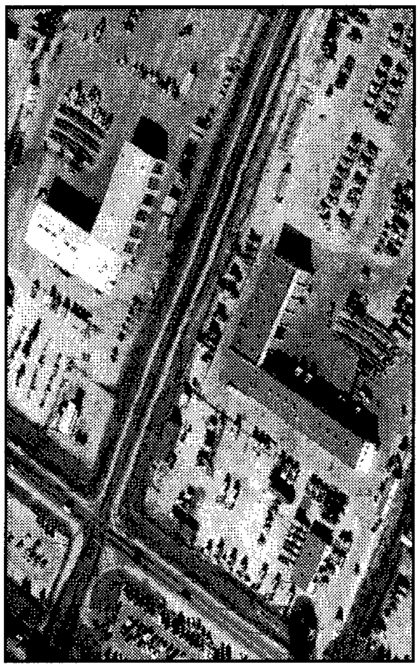


Figure B-225: RADT11WOB image

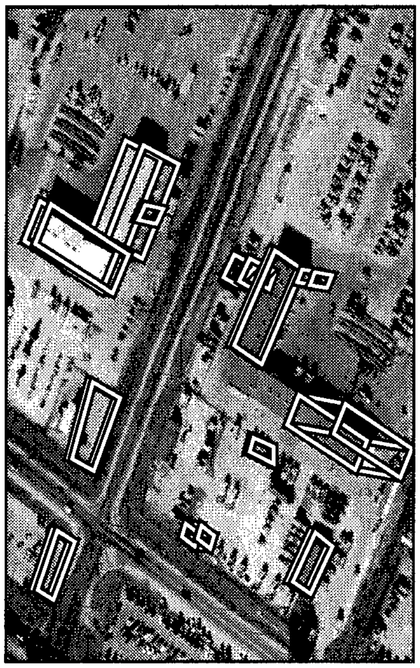


Figure B-226: BUILD+SHAVE results, RADT11WOB

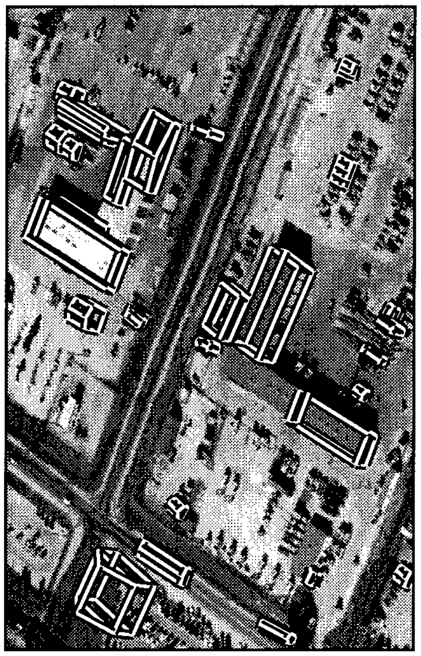


Figure B-227: VHBUILD results, RADT11WOB

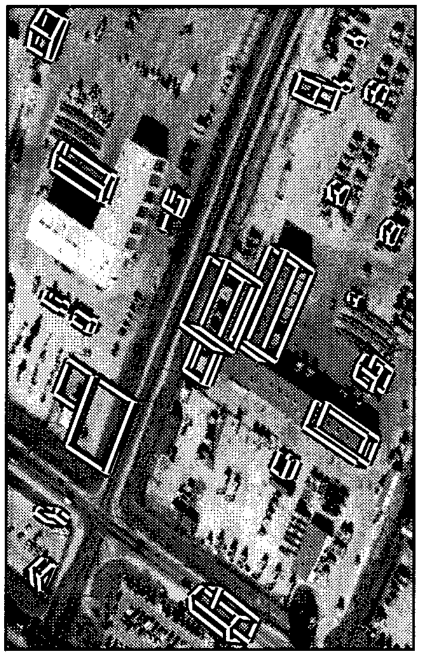


Figure B-228: PIVOT results, RADT11WOB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	69.75	**0.332**	**56.65**	-	-	-
BUILD+SHAVE	**81.13**	0.570	55.49	**52.96**	**1.456**	**29.90**
VHBUILD	69.56	1.065	39.96	50.94	1.480	29.04
PIVOT	34.67	2.899	17.29	29.60	2.627	16.65

Table B-61: 2D and 3D evaluation results for RADT11WOB

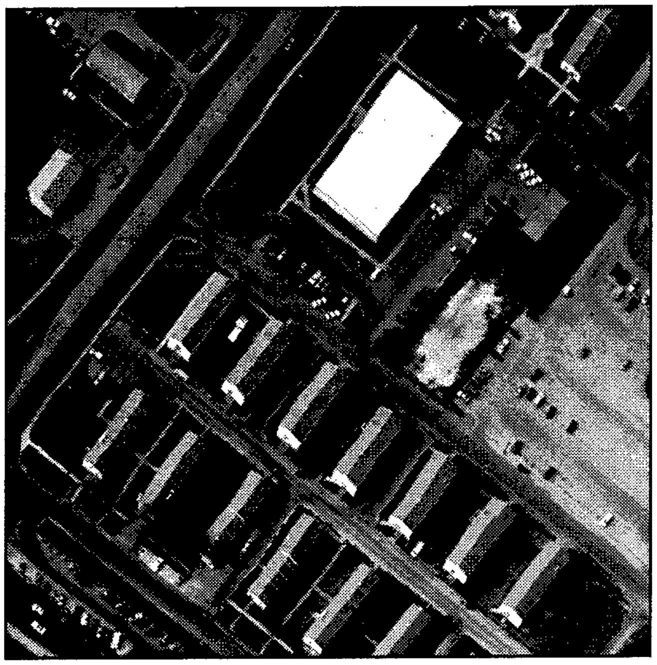


Figure B-229: RADT5 image

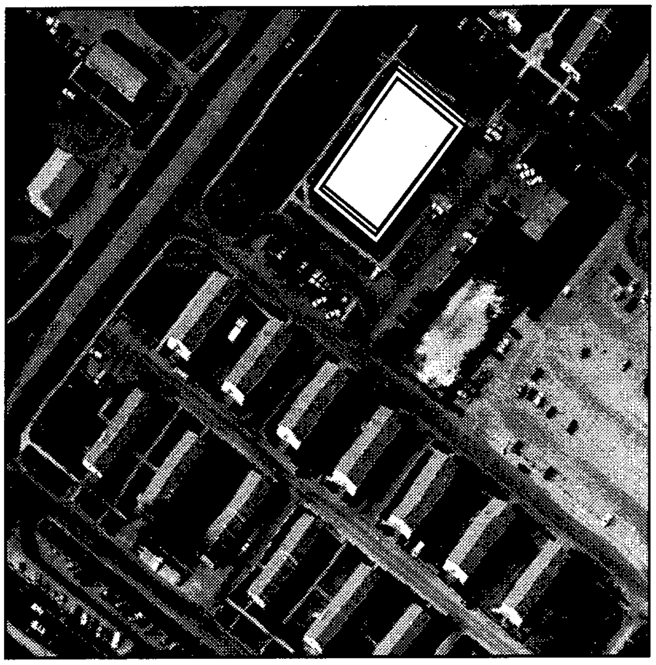


Figure B-230: BUILD+SHAVE results, RADT5

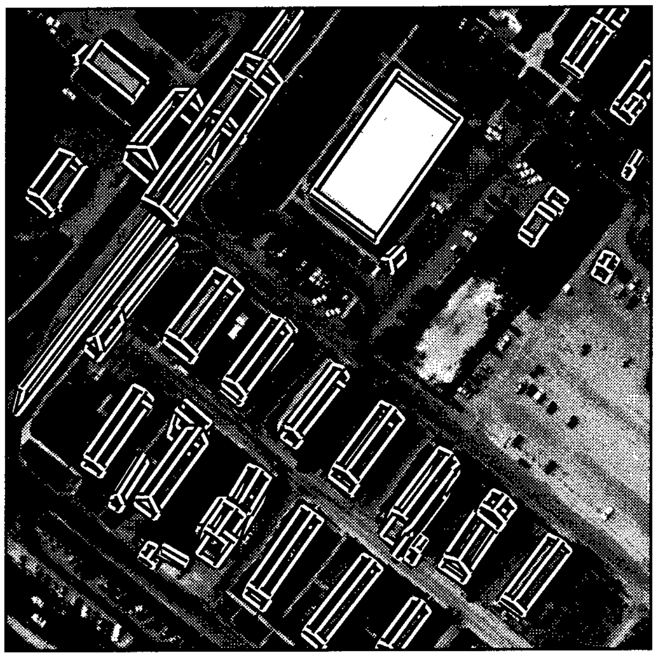


Figure B-231: VHBUILD results, RADT5

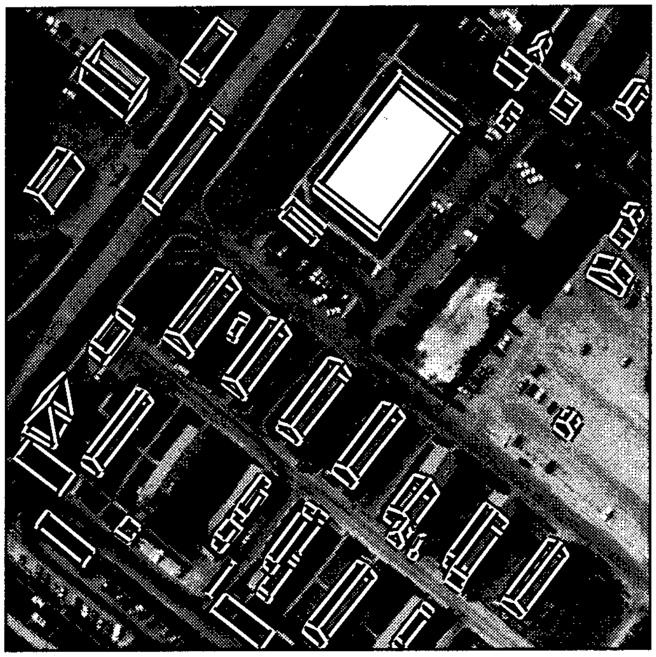


Figure B-232: PIVOT results, RADT5

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	21.59	**0.000**	21.59	-	-	-
BUILD+SHAVE	22.61	**0.000**	22.61	15.44	**0.025**	15.38
VHBUILD	**93.57**	0.656	**57.97**	**73.63**	1.948	30.24
PIVOT	76.04	0.600	52.21	68.52	0.890	**42.56**

Table B-62: 2D and 3D evaluation results for RADT5

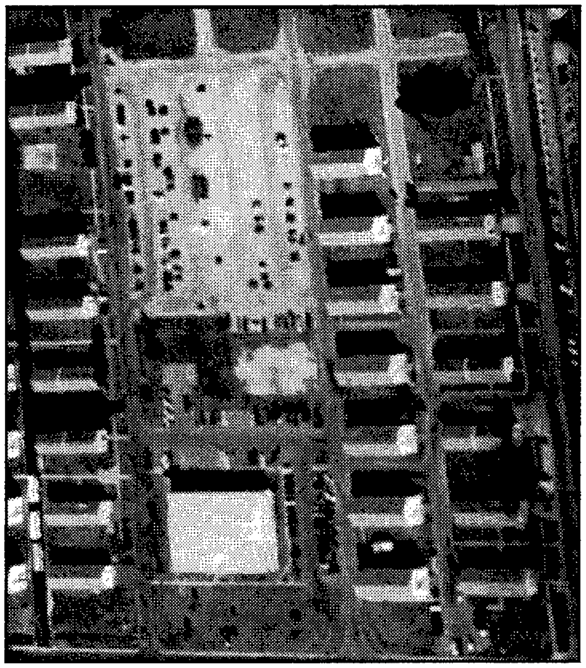


Figure B-233: RADT5OB image

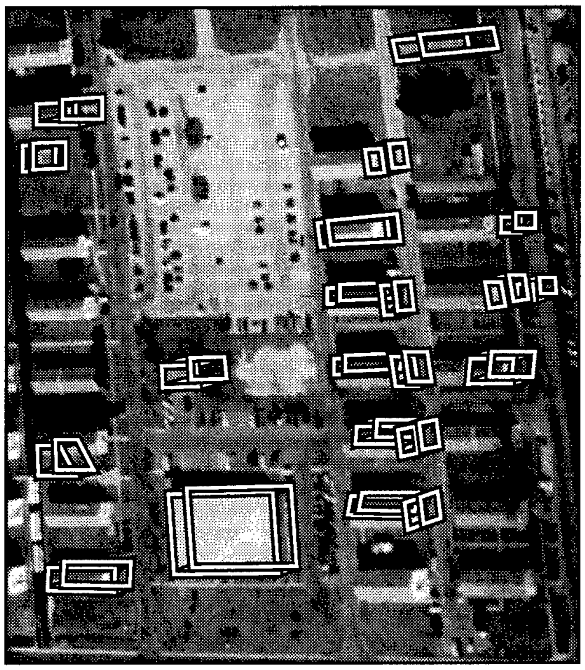


Figure B-234: BUILD+SHAVE results, RADT5OB

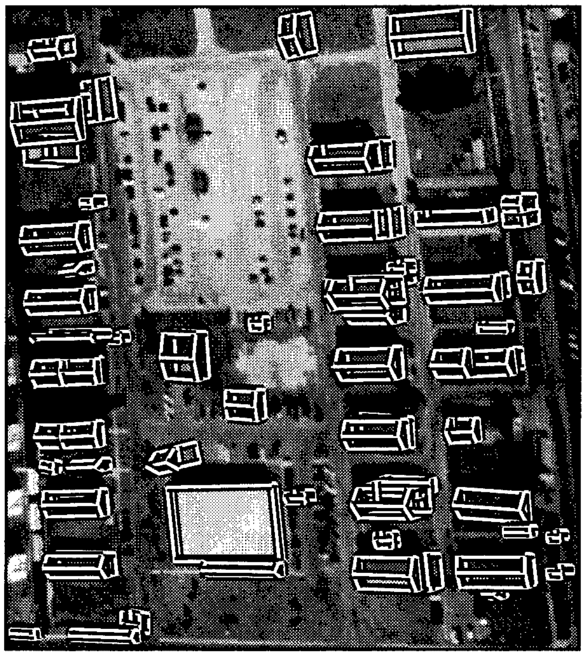


Figure B-235: VHBUILD results, RADT5OB

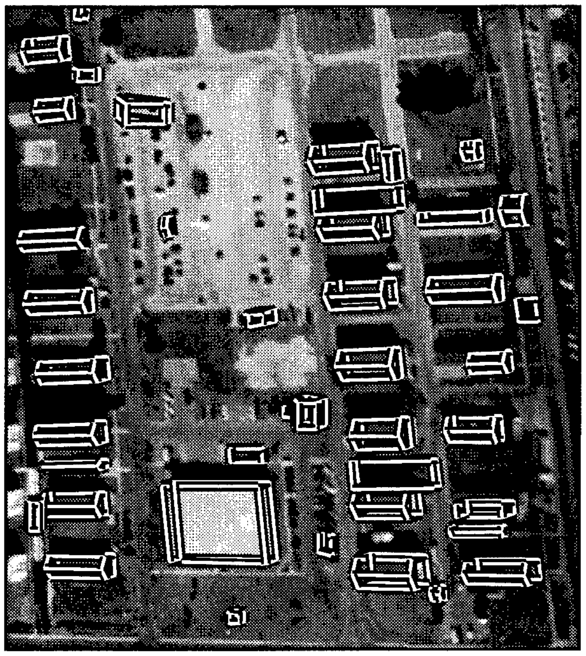


Figure B-236: PIVOT results, RADT5OB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	40.90	**0.281**	36.69	-	-	-
BUILD+SHAVE	43.21	0.813	31.98	32.06	2.445	17.97
VHBUILD	**91.74**	0.757	54.14	70.71	1.323	36.53
PIVOT	85.14	0.395	**63.70**	**76.90**	**0.526**	**54.76**

Table B-63: 2D and 3D evaluation results for RADT5OB

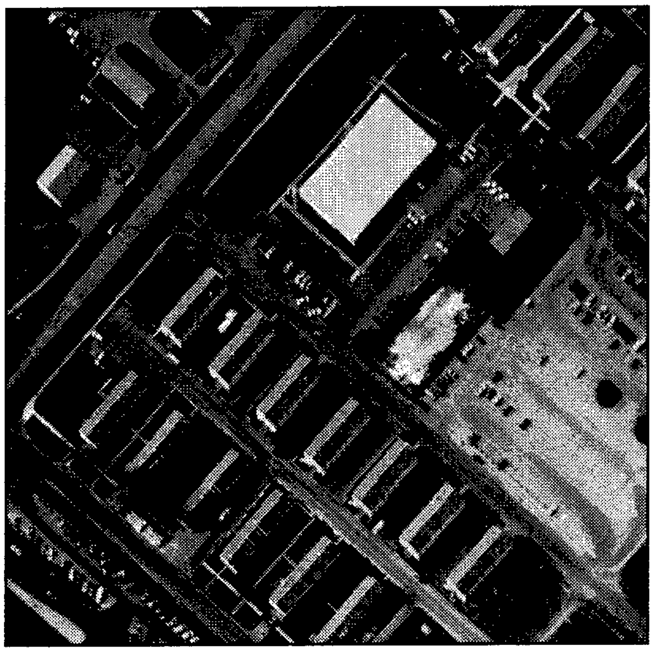


Figure B-237: RADT5S image

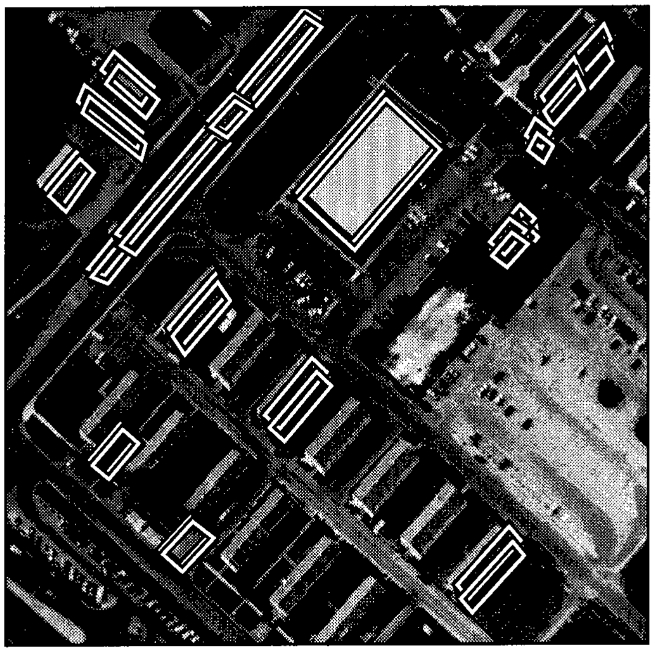


Figure B-238: BUILD+SHAVE results, RADT5S

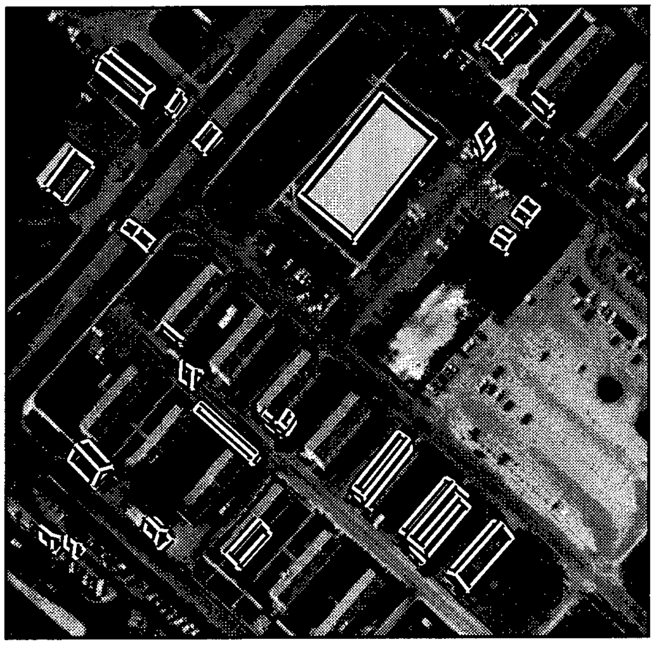


Figure B-239: VHBUILD results, RADT5S

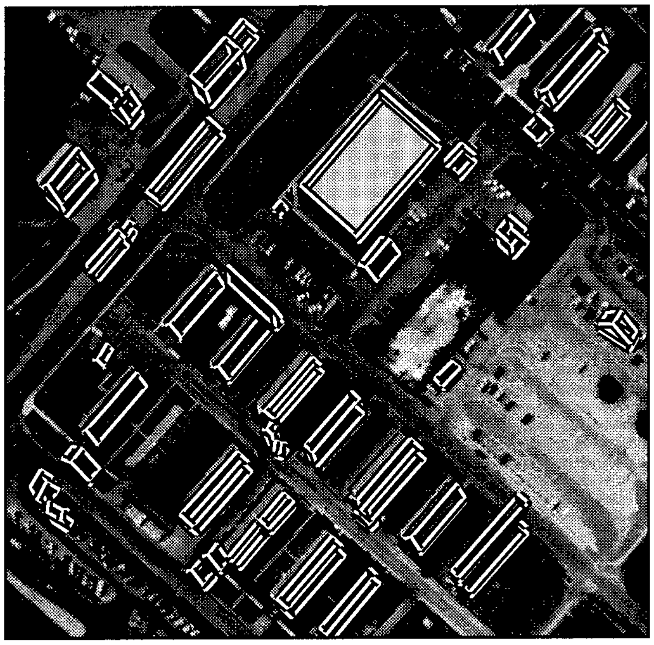


Figure B-240: PIVOT results, RADT5S

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	35.40	0.620	29.03	-	-	-
BUILD+SHAVE	38.86	1.040	27.67	30.71	1.979	19.10
VHBUILD	41.85	**0.395**	35.91	26.79	**0.704**	22.54
PIVOT	**65.96**	0.644	**46.30**	**51.27**	1.143	**32.33**

Table B-64: 2D and 3D evaluation results for RADT5S

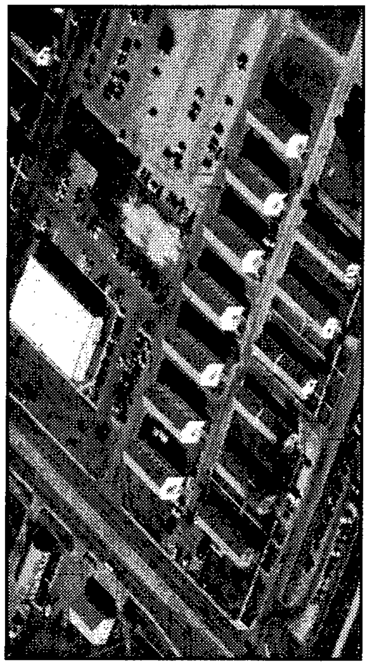


Figure B-241: RADT5WOB image

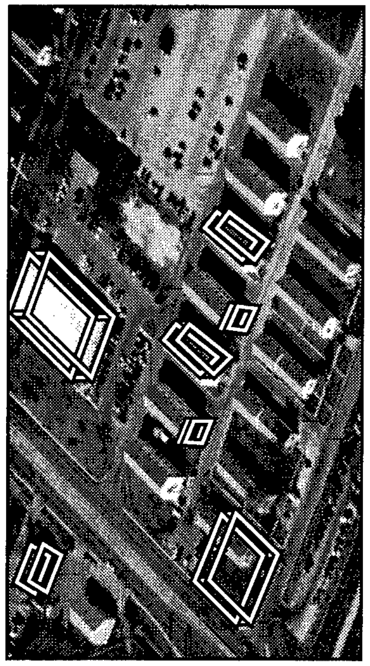


Figure B-242: BUILD+SHAVE results, RADT5WOB

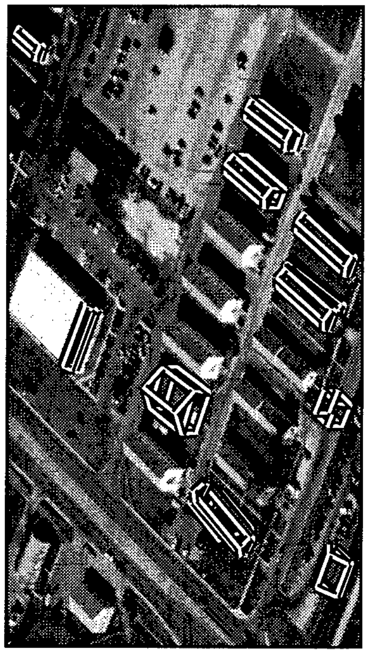


Figure B-243: VHBUILD results, RADT5WOB

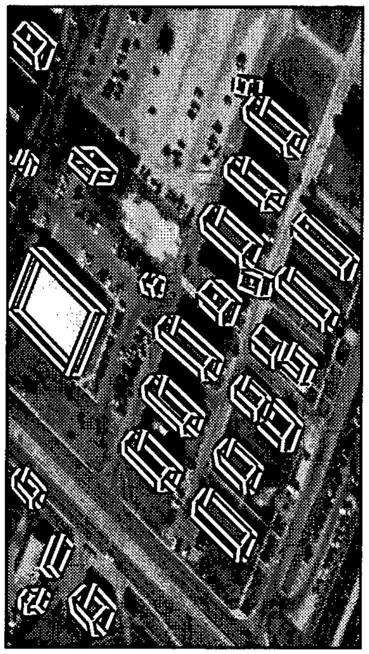


Figure B-244: PIVOT results, RADT5WOB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	26.90	0.270	25.08	-	-	-
BUILD+SHAVE	33.42	0.507	28.58	29.74	1.224	21.81
VHBUILD	31.22	0.371	27.98	20.77	0.788	17.85
PIVOT	**84.25**	**0.256**	**69.28**	**76.97**	**0.292**	**62.84**

Table B-65: 2D and 3D evaluation results for RADT5WOB

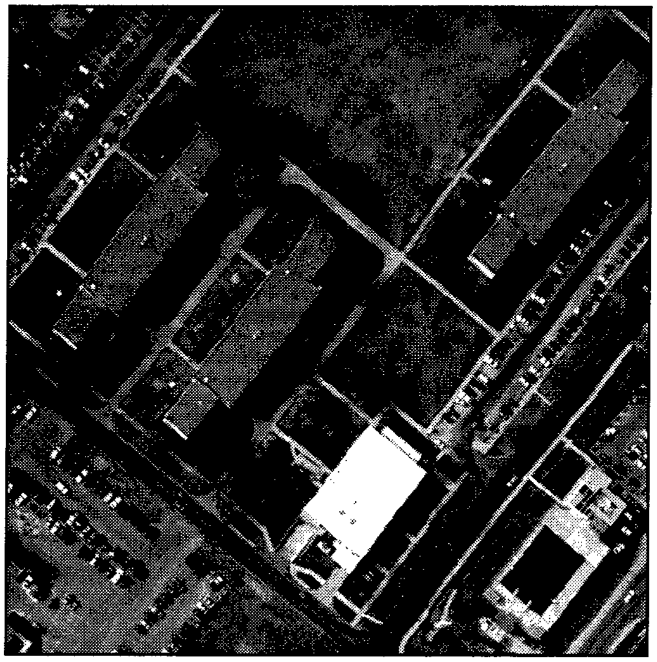


Figure B-245: RADT6 image

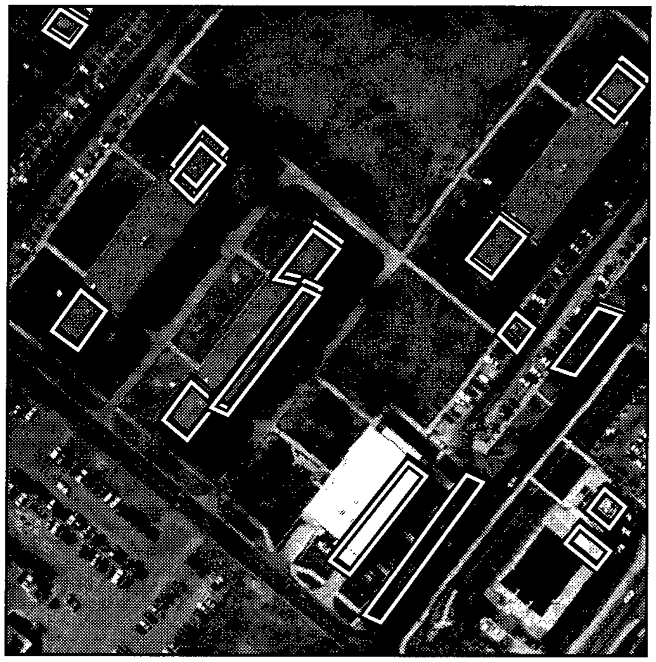


Figure B-246: BUILD+SHAVE results, RADT6

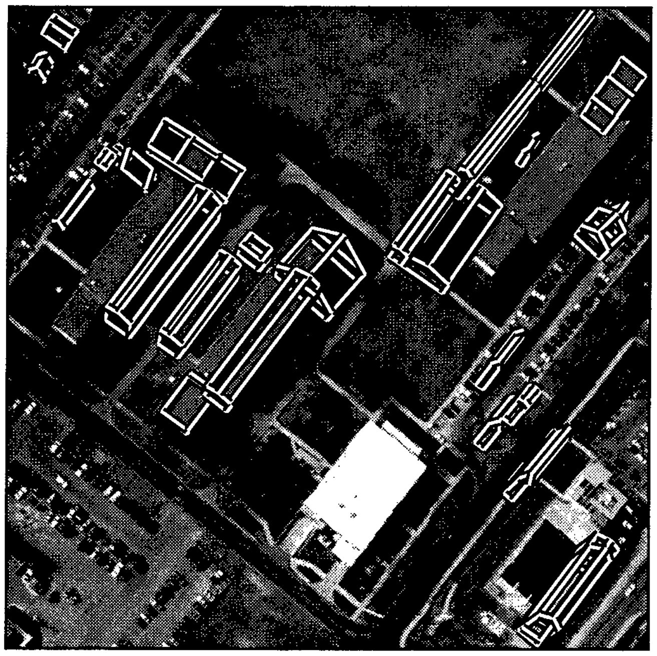


Figure B-247: VHBUILD results, RADT6

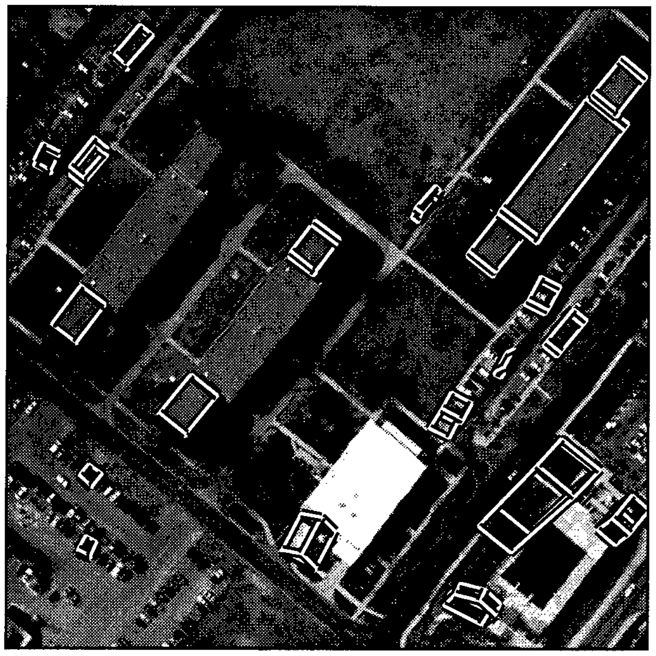


Figure B-248: PIVOT results, RADT6

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	36.13	**0.551**	30.13	-	-	-
BUILD+SHAVE	**40.78**	0.559	**33.21**	32.94	**1.166**	**23.80**
VHBUILD	32.98	2.574	17.84	27.87	10.644	7.03
PIVOT	36.35	1.128	25.78	**32.99**	3.610	15.06

Table B-66: 2D and 3D evaluation results for RADT6

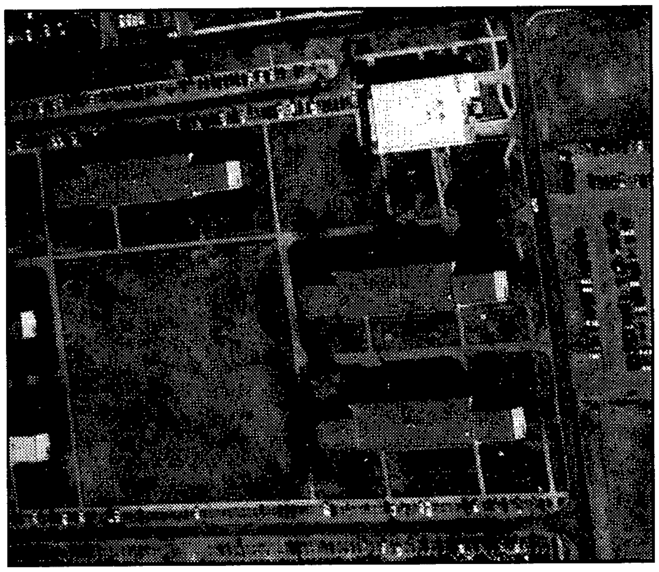


Figure B-249: RADT6OB image

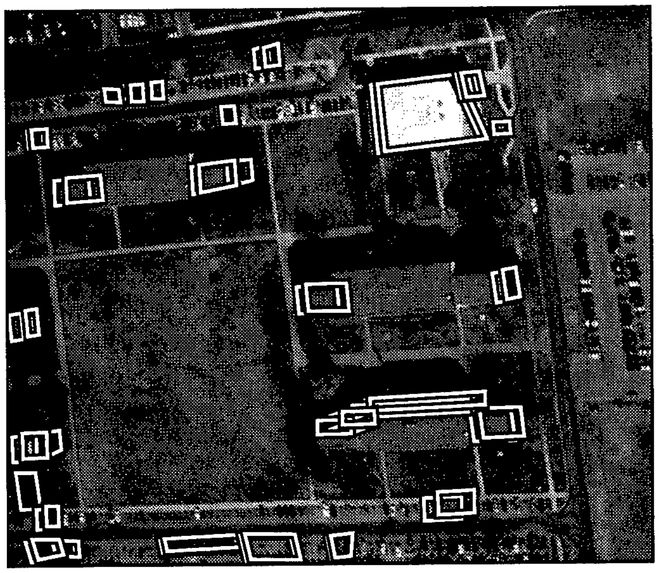


Figure B-250: BUILD+SHAVE results, RADT6OB

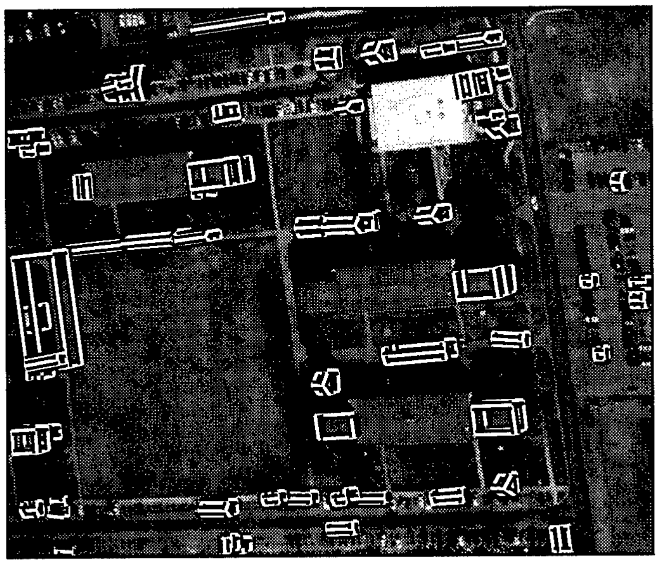


Figure B-251: VHBUILD results, RADT6OB

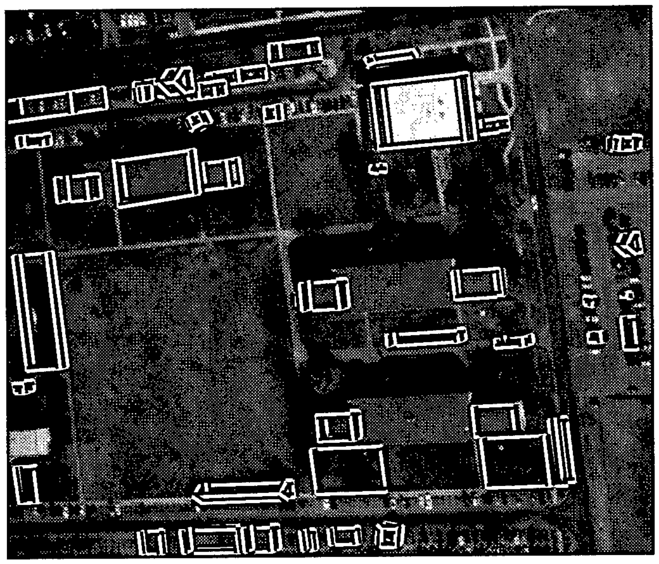


Figure B-252: PIVOT results, RADT6OB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	42.22	**0.617**	33.50	-	-	-
BUILD+SHAVE	49.59	0.811	**35.37**	29.06	**1.116**	**21.94**
VHBUILD	20.46	4.410	10.76	12.53	6.301	7.00
PIVOT	**56.08**	1.850	27.53	**40.65**	2.507	20.14

Table B-67: 2D and 3D evaluation results for RADT6OB

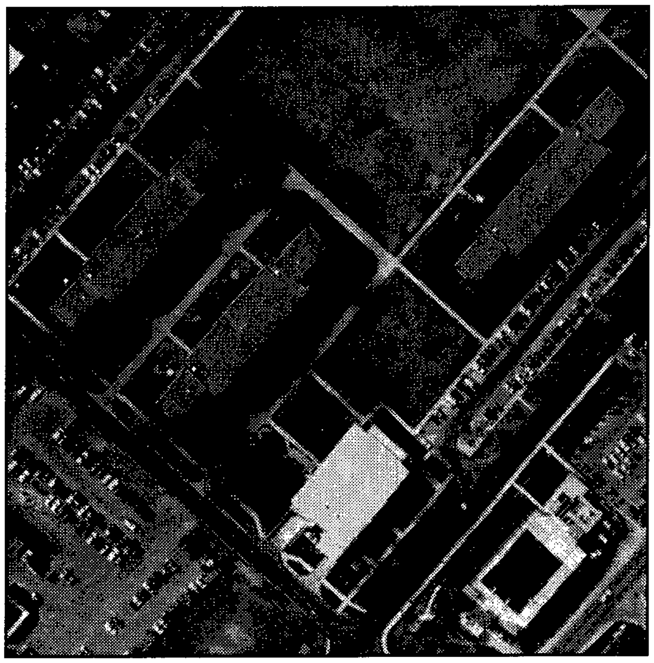


Figure B-253: RADT6S image

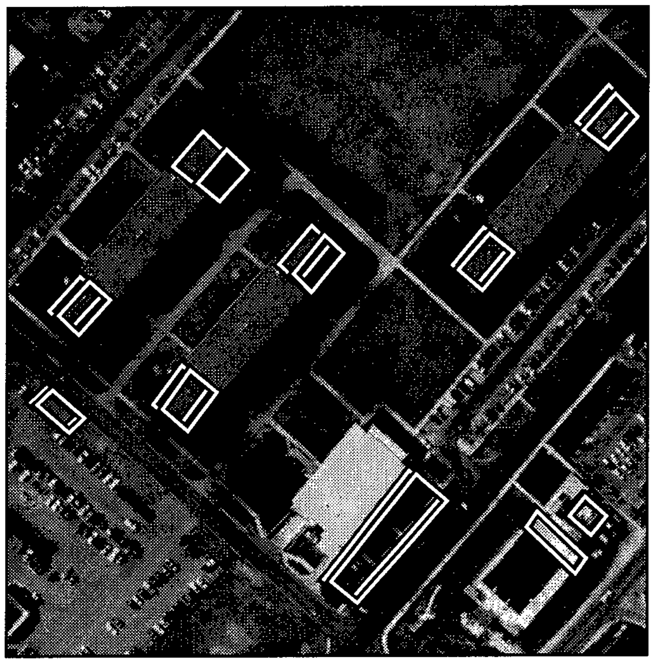


Figure B-254: BUILD+SHAVE results, RADT6S

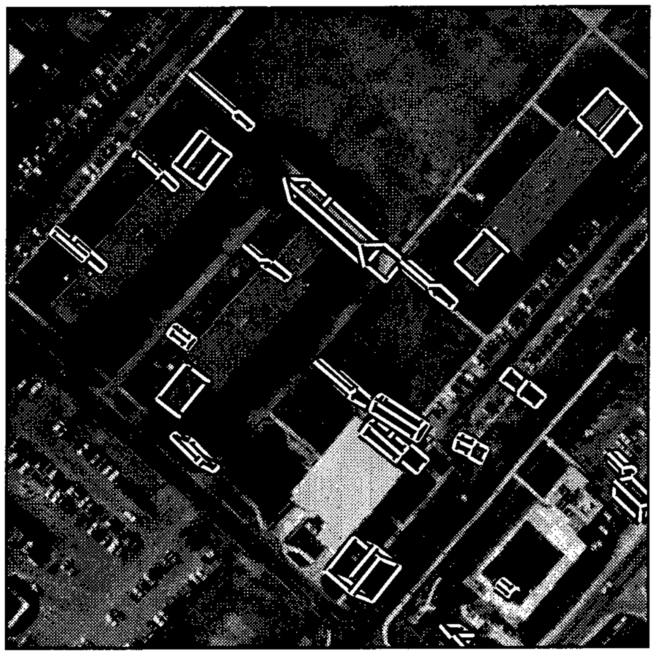


Figure B-255: VHBUILD results, RADT6S

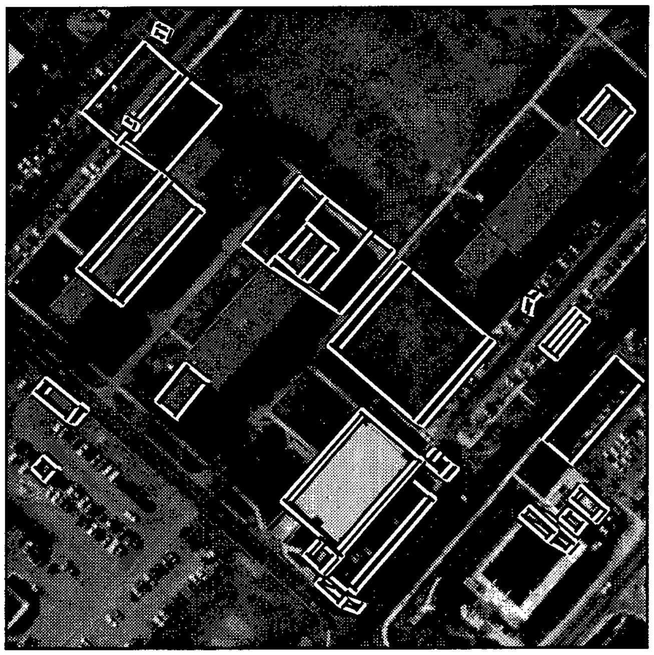


Figure B-256: PIVOT results, RADT6S

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	20.69	**0.917**	17.39	-	-	-
BUILD+SHAVE	26.02	1.016	20.58	19.78	**2.125**	**13.93**
VHBUILD	19.07	2.074	13.67	9.50	11.024	4.64
PIVOT	**50.56**	2.303	**23.36**	**39.15**	6.399	11.17

Table B-68: 2D and 3D evaluation results for RADT6S

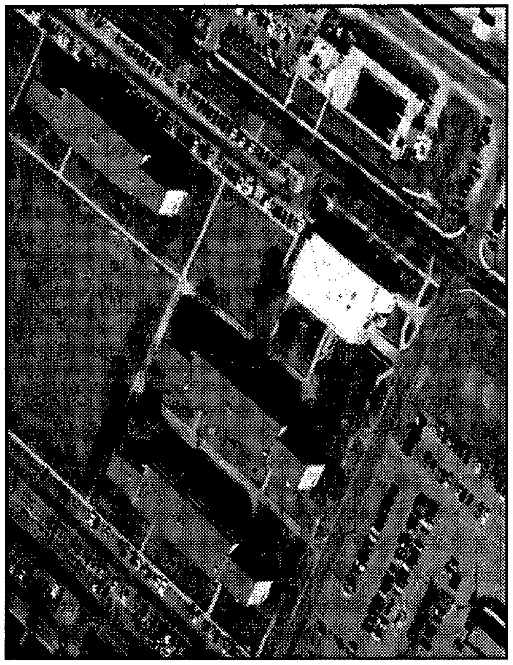


Figure B-257: RADT6WOB image

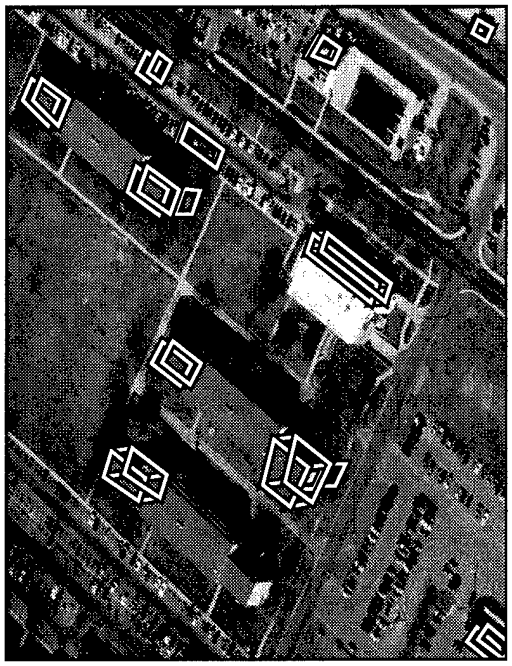


Figure B-258: BUILD+SHAVE results, RADT6WOB

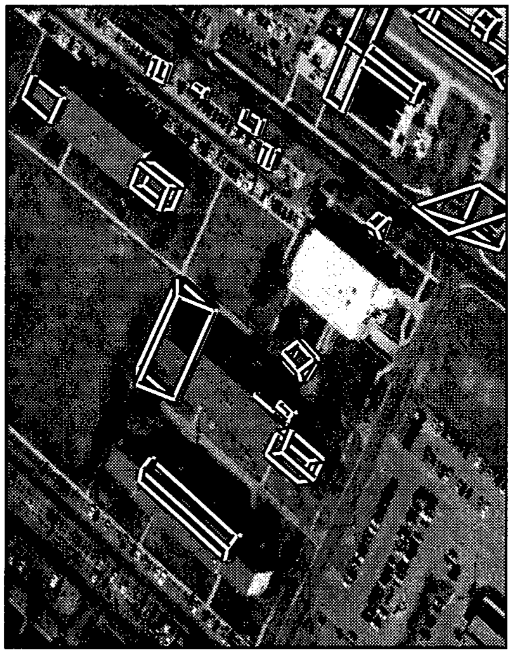


Figure B-259: VHBUILD results, RADT6WOB

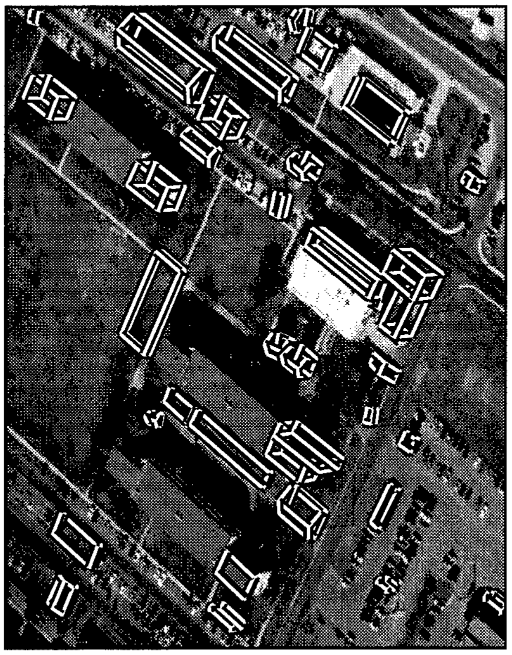


Figure B-260: PIVOT results, RADT6WOB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	20.27	**0.474**	18.49	-	-	-
BUILD+SHAVE	**30.75**	0.557	**26.25**	**16.76**	**1.181**	**13.99**
VHBUILD	28.34	1.901	18.42	12.73	3.542	8.77
PIVOT	25.88	3.658	13.29	14.97	3.784	9.55

Table B-69: 2D and 3D evaluation results for RADT6WOB

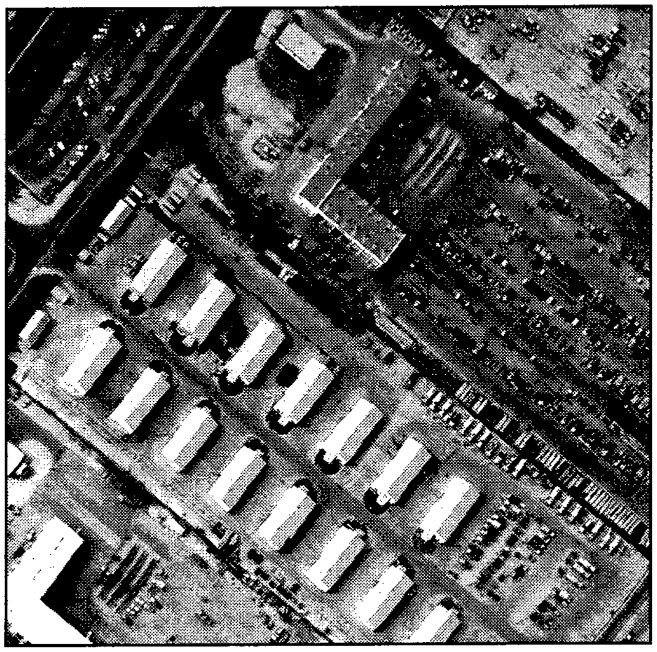


Figure B-261: RADT9 image

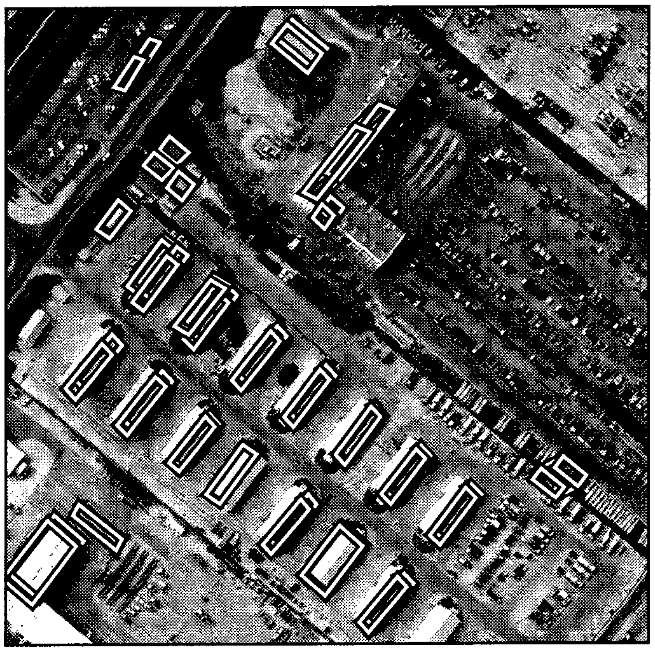


Figure B-262: BUILD+SHAVE results, RADT9

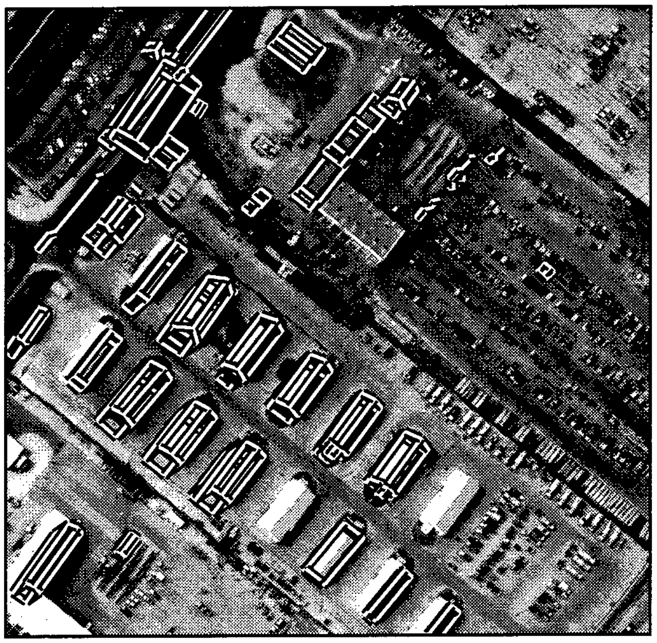


Figure B-263: VHBUILD results, RADT9

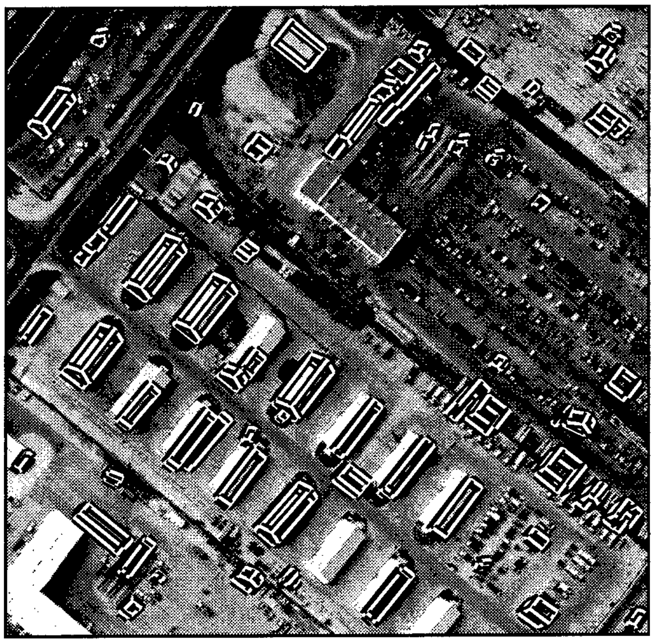


Figure B-264: PIVOT results, RADT9

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	45.99	**0.189**	42.31	-	-	-
BUILD+SHAVE	55.17	0.281	**47.77**	43.65	**1.121**	**29.31**
VHBUILD	**61.12**	0.668	43.40	**46.53**	3.422	17.95
PIVOT	43.86	1.085	29.72	34.40	1.817	21.17

Table B-70: 2D and 3D evaluation results for RADT9

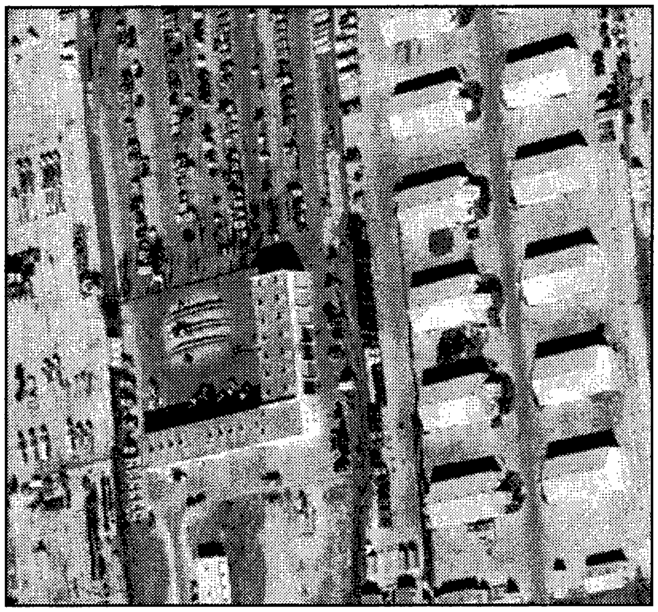


Figure B-265: RADT9OB image

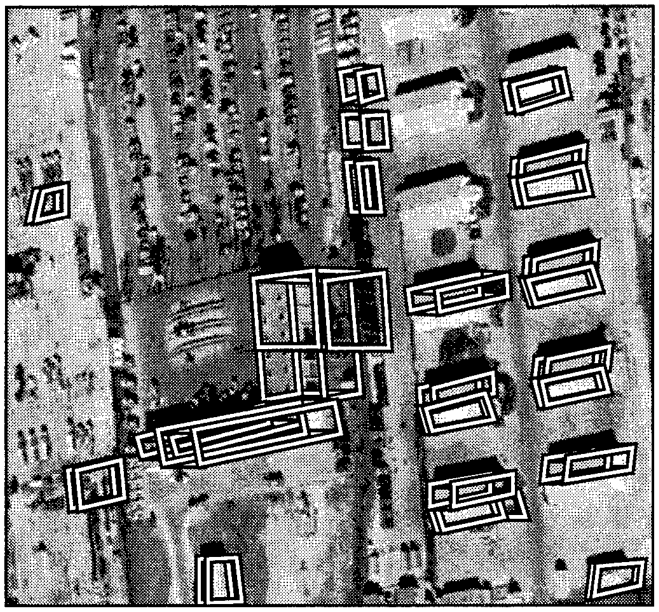


Figure B-266: BUILD+SHAVE results, RADT9OB

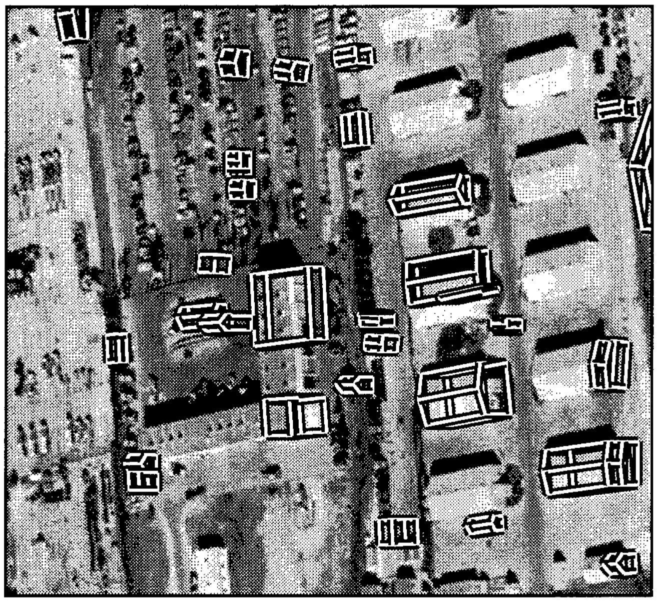


Figure B-267: VHBUILD results, RADT9OB

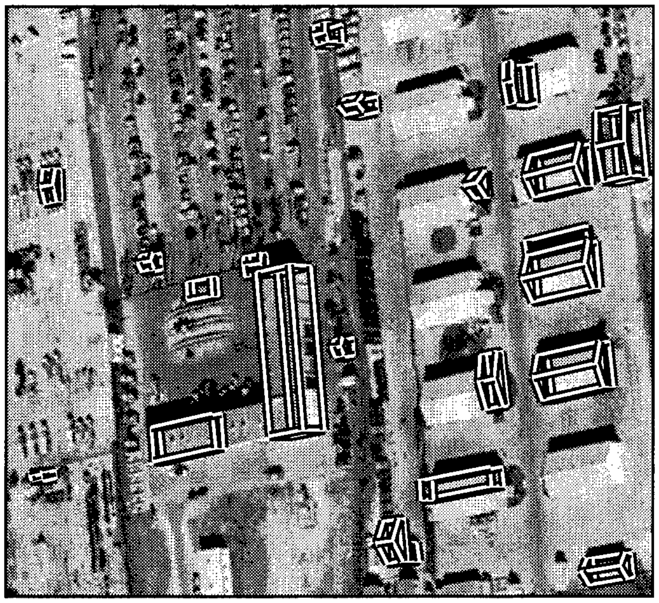


Figure B-268: PIVOT results, RADT9OB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	56.79	**0.210**	50.73	-	-	-
BUILD+SHAVE	**68.07**	0.500	**50.79**	**38.09**	3.217	17.12
VHBUILD	33.49	1.141	24.23	24.18	2.157	15.89
PIVOT	46.53	0.484	37.97	37.36	**0.561**	**30.88**

Table B-71: 2D and 3D evaluation results for RADT9OB

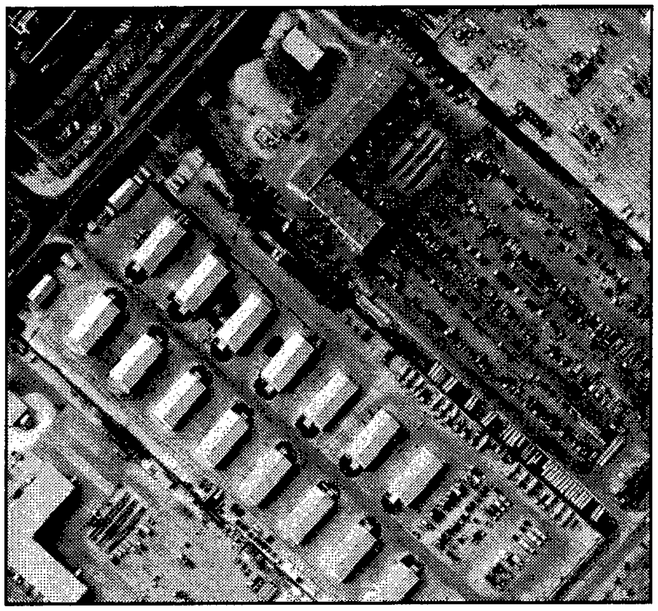


Figure B-269: RADT9S image

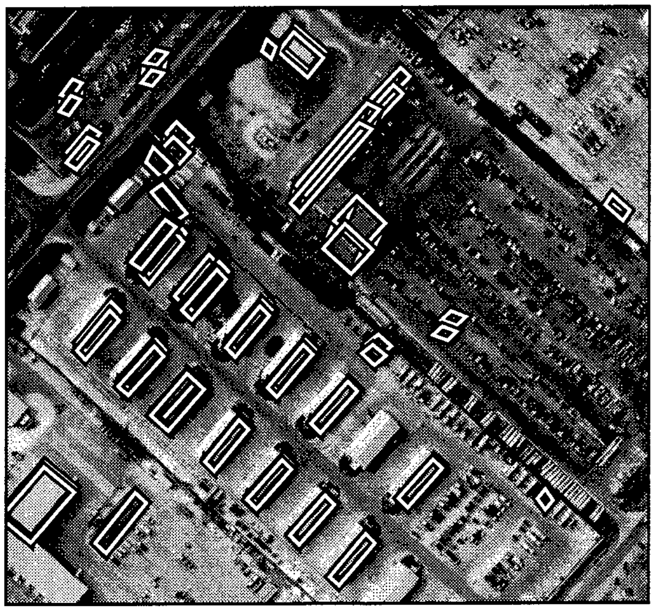


Figure B-270: BUILD+SHAVE results, RADT9S

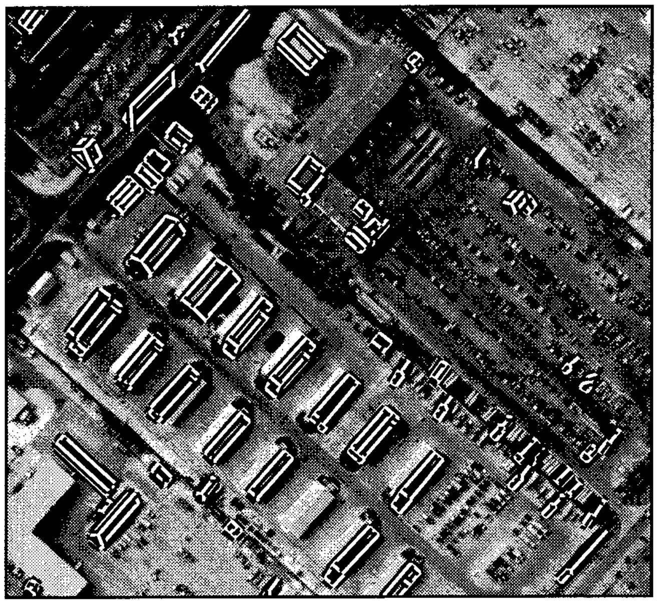


Figure B-271: VHBUILD results, RADT9S

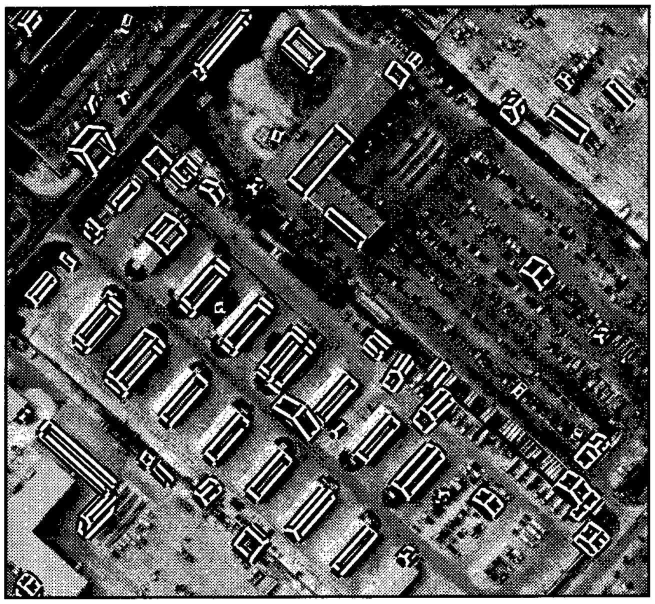


Figure B-272: PIVOT results, RADT9S

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	49.70	**0.268**	**43.85**	-	-	-
BUILD+SHAVE	**54.15**	0.549	41.74	**38.30**	**2.358**	**20.13**
VHBUILD	42.22	0.990	29.77	28.89	3.273	14.85
PIVOT	46.06	1.140	30.21	30.40	2.996	15.91

Table B-72: 2D and 3D evaluation results for RADT9S

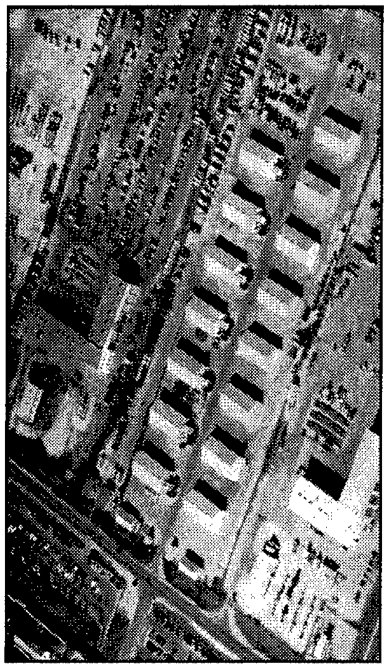


Figure B-273: RADT9WOB image

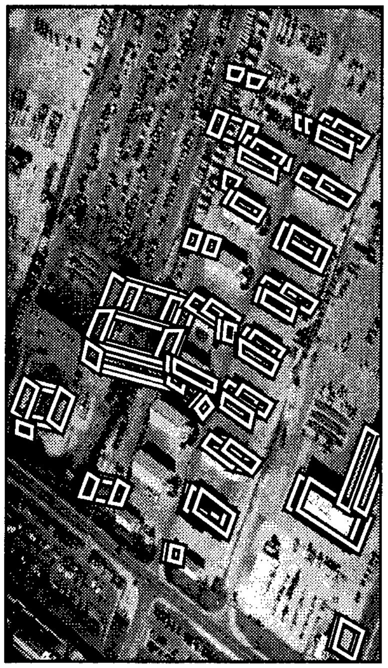


Figure B-274: BUILD+SHAVE results, RADT9WOB

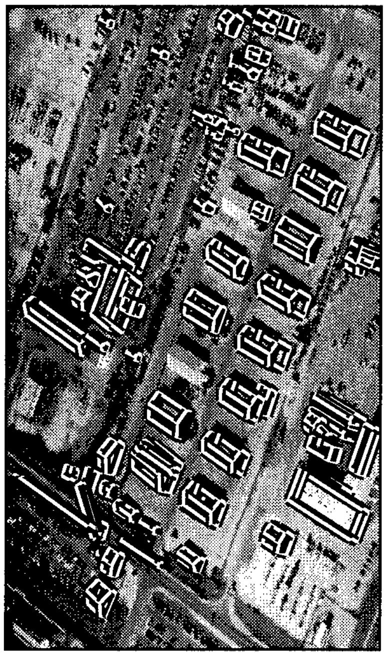


Figure B-275: VHBUILD results, RADT9WOB

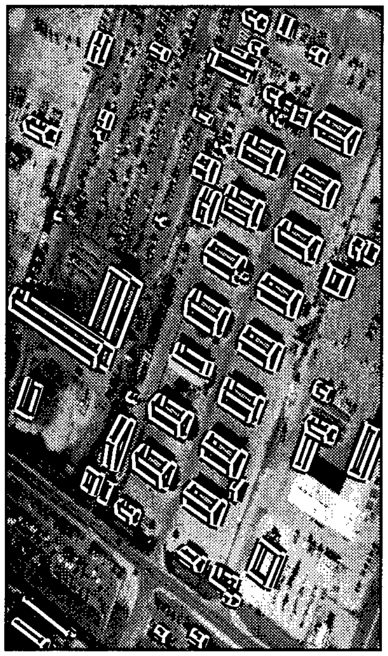


Figure B-276: PIVOT results, RADT9WOB

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	53.45	**0.224**	47.74	-	-	-
BUILD+SHAVE	68.58	0.794	44.39	35.89	3.577	15.71
VHBUILD	76.17	0.796	47.42	52.87	1.426	30.14
PIVOT	**80.48**	0.682	**51.95**	**72.02**	**0.706**	**47.74**

Table B-73: 2D and 3D evaluation results for RADT9WOB

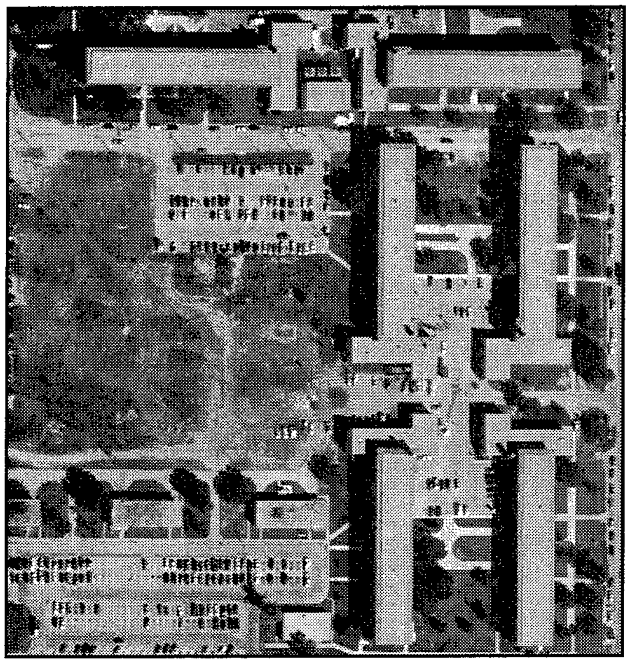


Figure B-277: T_FHN711 image

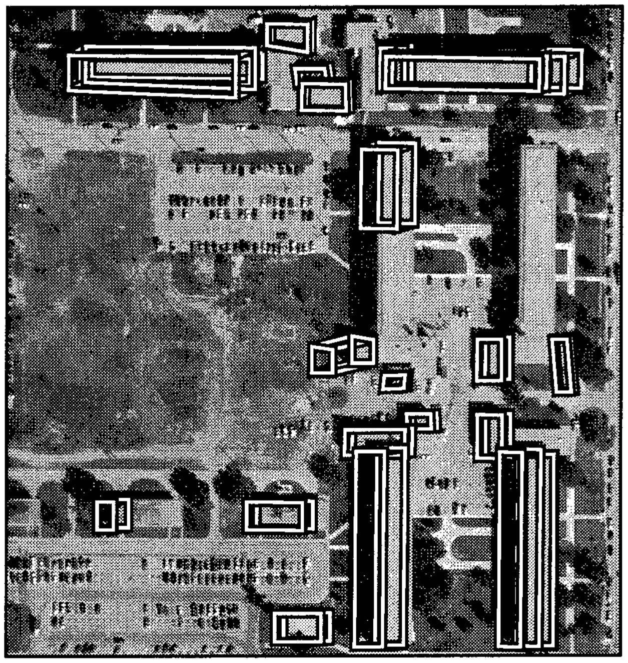


Figure B-278: BUILD+SHAVE results, T_FHN711

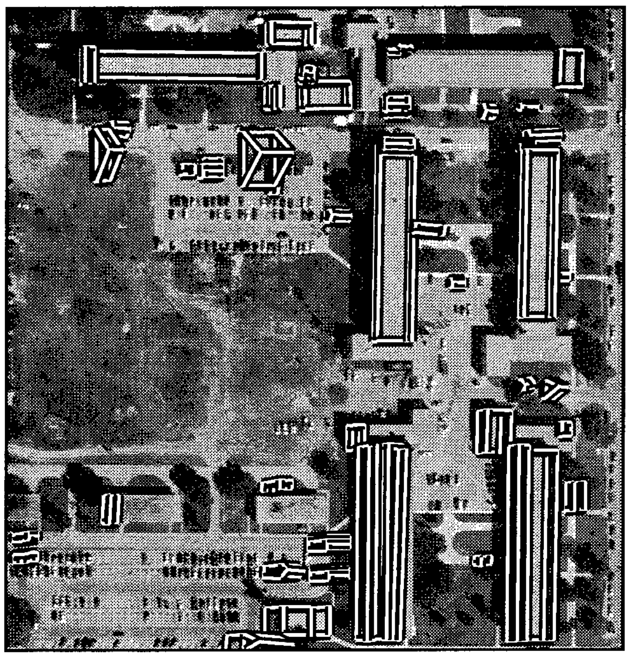


Figure B-279: VHBUILD results, T_FHN711

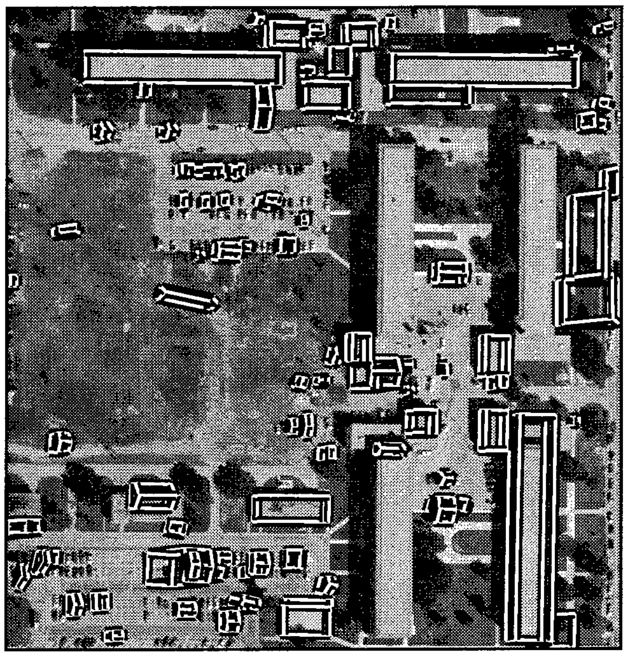


Figure B-280: PIVOT results, T_FHN711

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	53.70	**0.049**	52.33	-	-	-
BUILD+SHAVE	**62.89**	0.255	**54.21**	**43.39**	1.768	24.55
VHBUILD	58.10	0.363	47.98	36.15	**1.101**	**25.86**
PIVOT	47.24	0.840	33.82	25.29	2.054	16.64

Table B-74: 2D and 3D evaluation results for T_FHN711

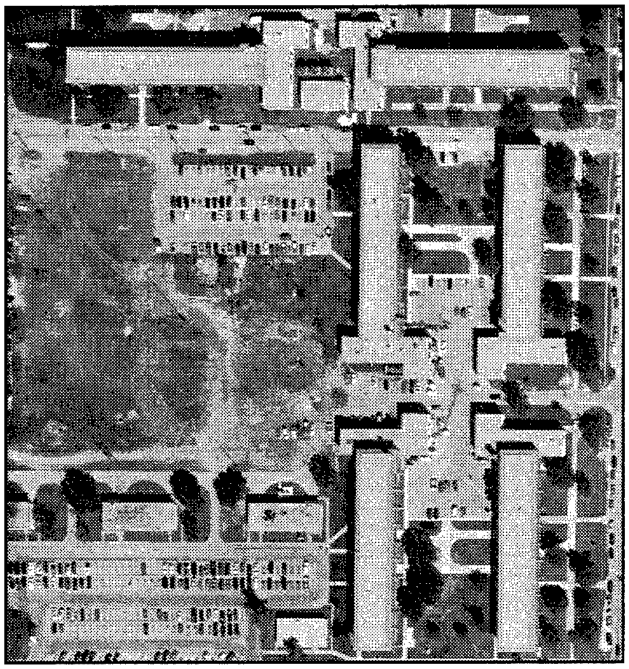


Figure B-281: T_FHN78 image

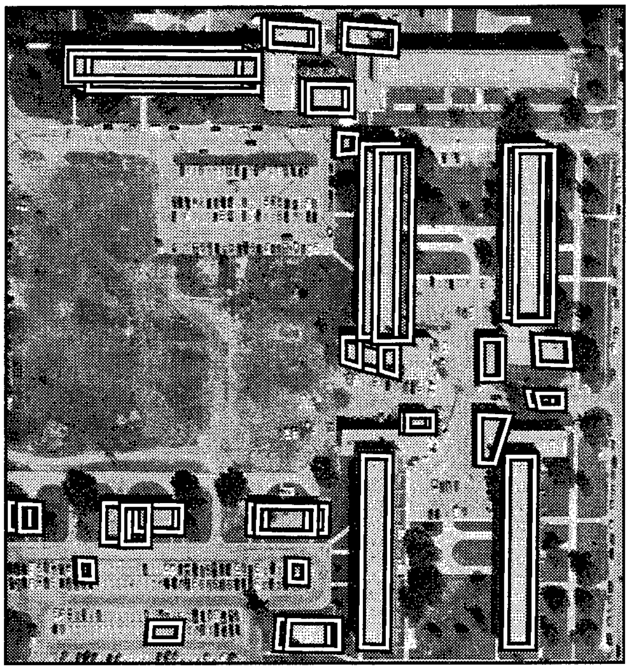


Figure B-282: BUILD+SHAVE results, T_FHN78

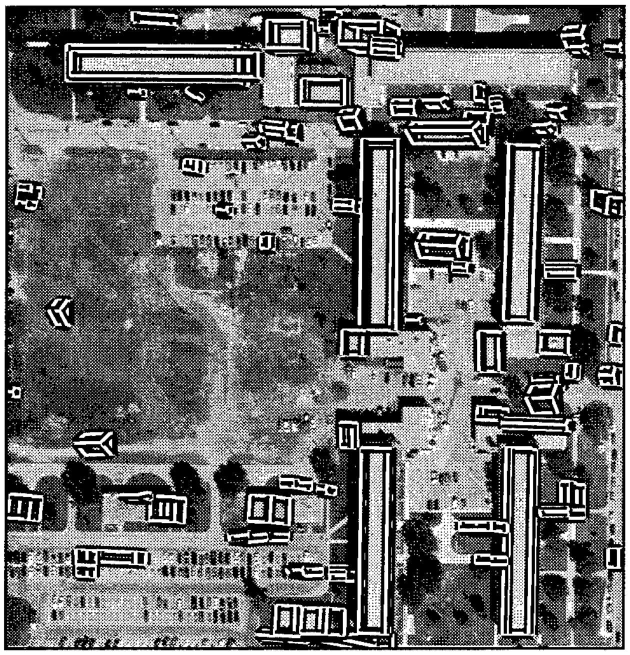


Figure B-283: VHBUILD results, T_FHN78

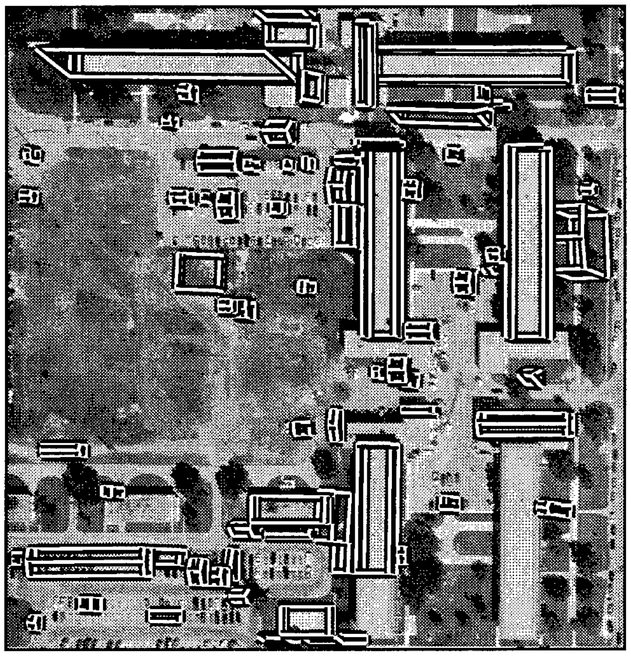


Figure B-284: PIVOT results, T_FHN78

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	60.06	**0.064**	57.84	-	-	-
BUILD+SHAVE	**67.02**	0.173	**60.05**	43.15	**1.267**	27.89
VHBUILD	60.88	0.489	46.91	30.77	1.532	20.91
PIVOT	62.14	0.760	42.21	**60.98**	1.384	**33.08**

Table B-75: 2D and 3D evaluation results for T_FHN78

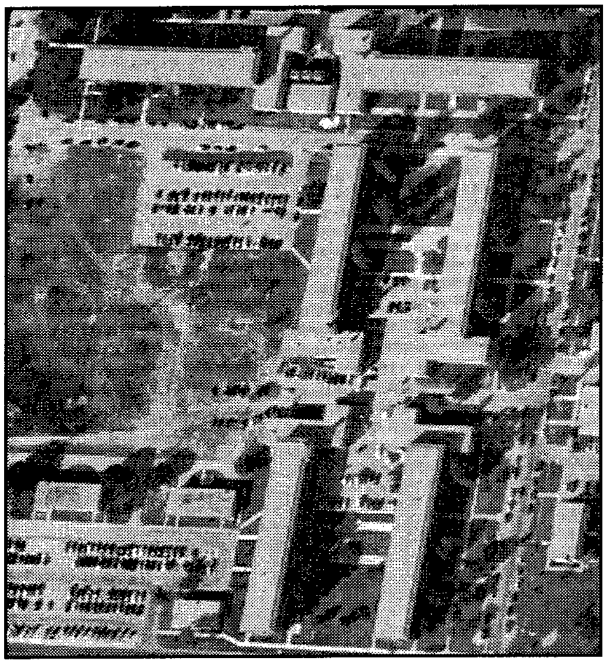


Figure B-285: T_FHOV1027 image

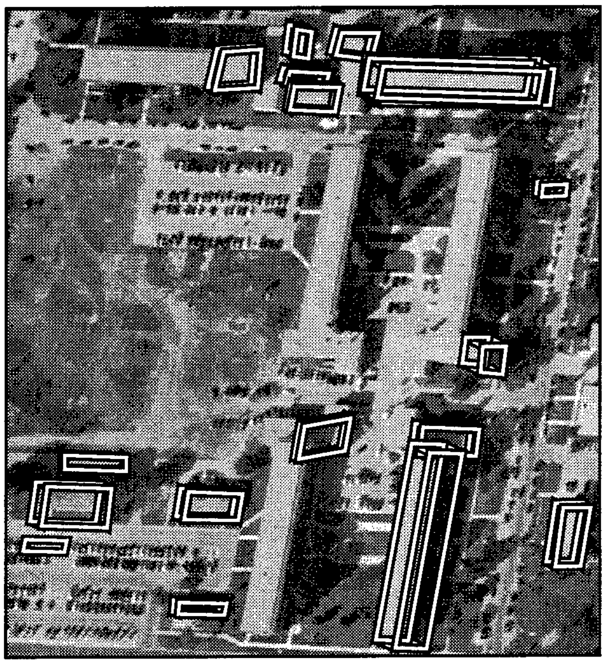


Figure B-286: BUILD+SHAVE results, T_FHOV1027

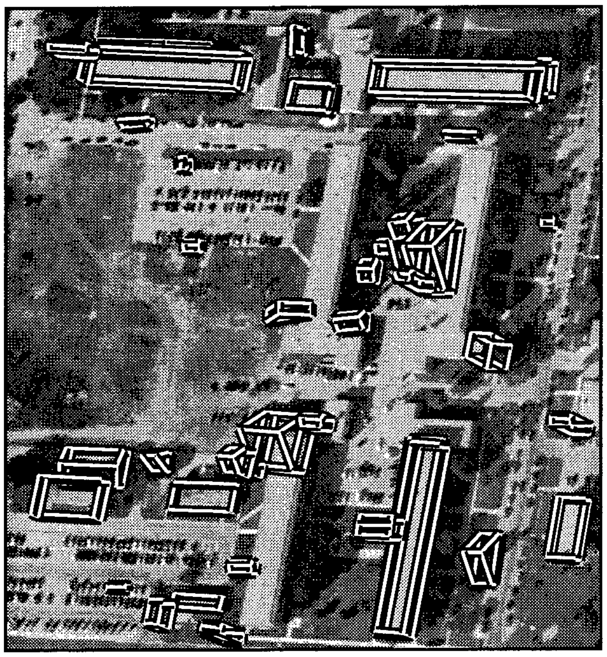


Figure B-287: VHBUILD results, T_FHOV1027

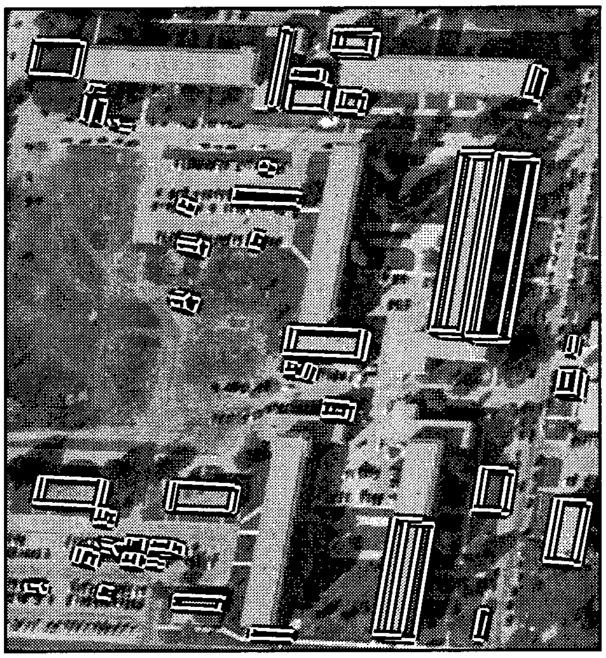


Figure B-288: PIVOT results, T_FHOV1027

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	35.60	**0.126**	34.08	-	-	-
BUILD+SHAVE	42.04	0.326	**36.97**	32.54	1.236	23.21
VHBUILD	**46.12**	0.634	35.68	**37.84**	1.131	**26.50**
PIVOT	37.16	0.713	29.38	26.21	**1.070**	20.47

Table B-76: 2D and 3D evaluation results for T_FHOV1027

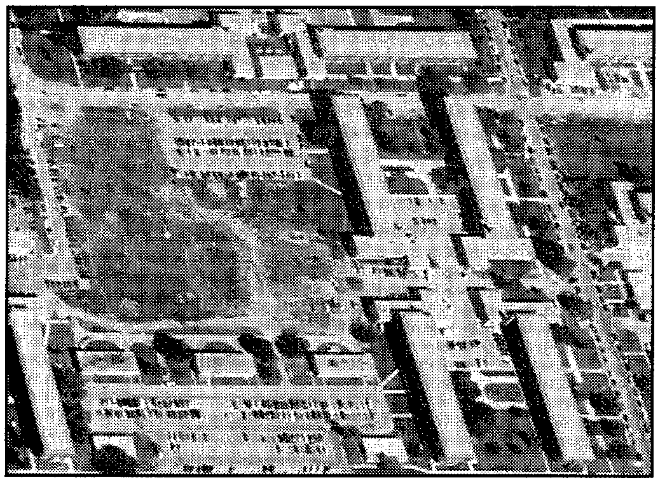


Figure B-289: T_FHOV525 image

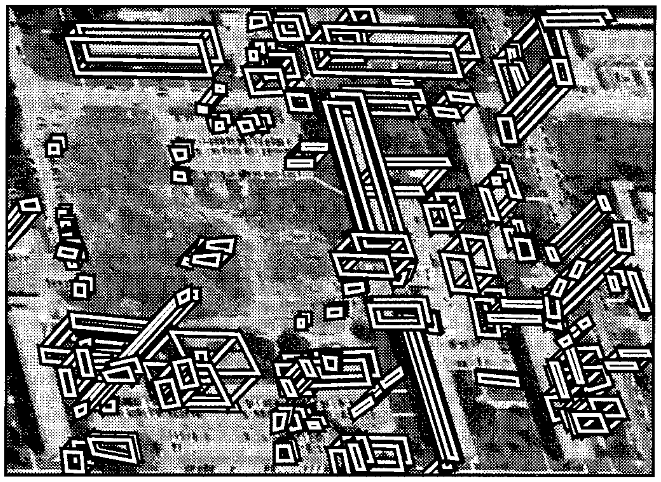


Figure B-290: BUILD+SHAVE results, T_FHOV525

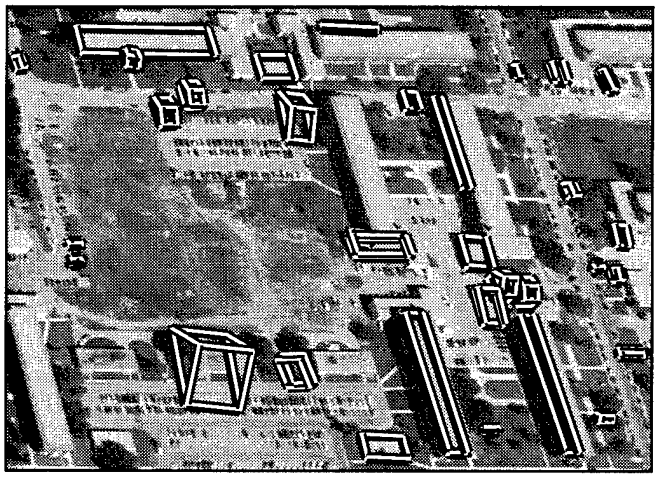


Figure B-291: VHBUILD results, T_FHOV525

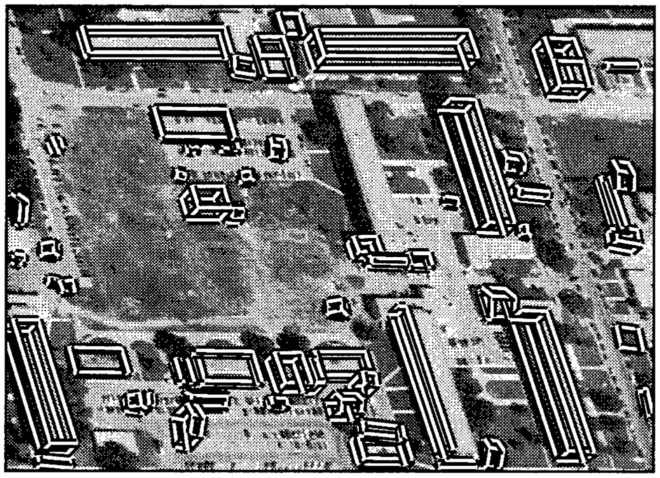


Figure B-292: PIVOT results, T_FHOV525

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	43.73	0.688	33.62	-	-	-
BUILD+SHAVE	60.25	1.487	31.78	35.41	5.079	12.65
VHBUILD	28.93	0.758	23.73	9.56	2.871	7.50
PIVOT	**64.73**	**0.642**	**45.72**	**50.55**	**1.101**	**32.48**

Table B-77: 2D and 3D evaluation results for T_FHOV525

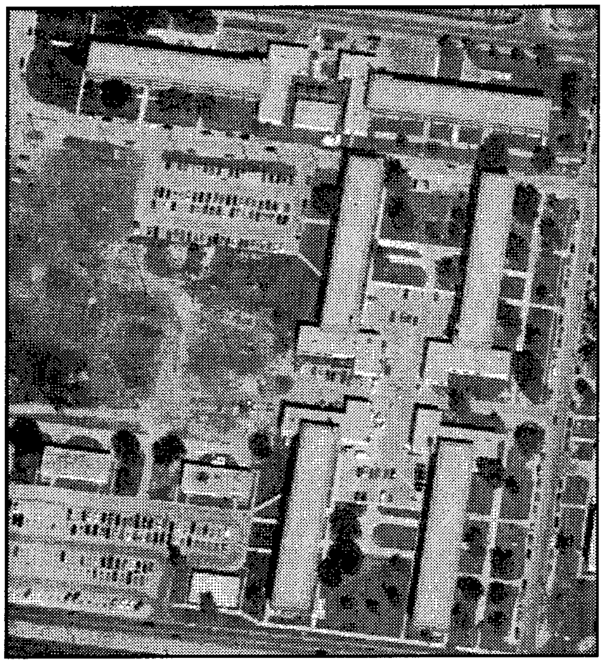


Figure B-293: T_FHOV625 image

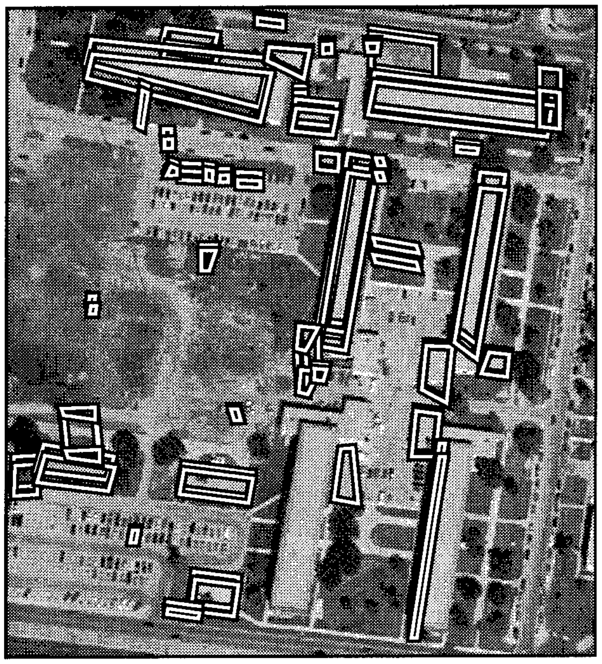


Figure B-294: BUILD+SHAVE results, T_FHOV625

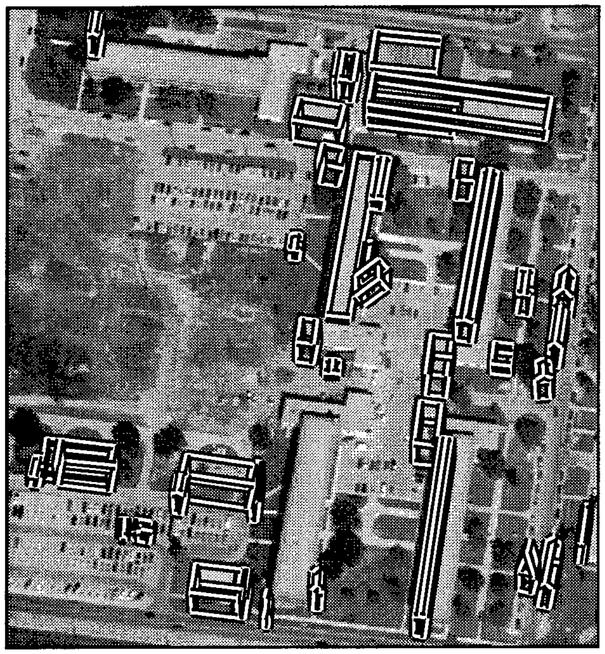


Figure B-295: VHBUILD results, T_FHOV625

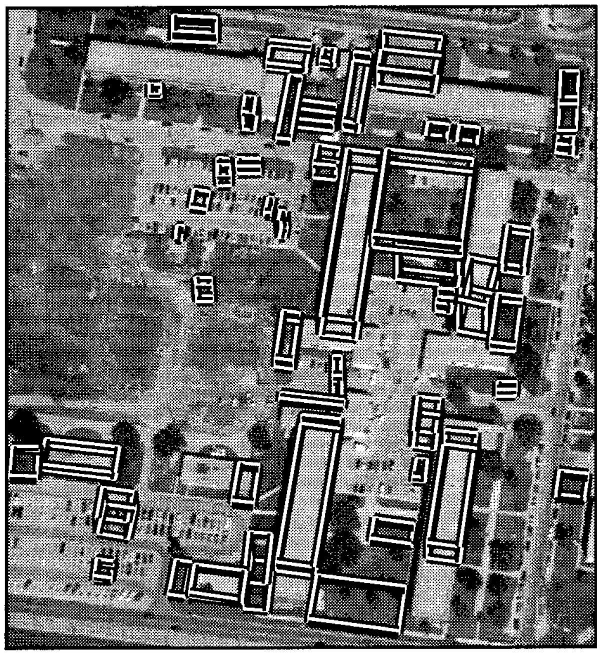


Figure B-296: PIVOT results, T_FHOV625

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	50.46	**0.348**	42.92	-	-	-
BUILD+SHAVE	**60.08**	0.510	**45.99**	**52.80**	**1.183**	**32.50**
VHBUILD	46.36	0.816	33.64	31.82	3.148	15.90
PIVOT	46.75	1.446	27.89	42.02	2.610	20.04

Table B-78: 2D and 3D evaluation results for T_FHOV625

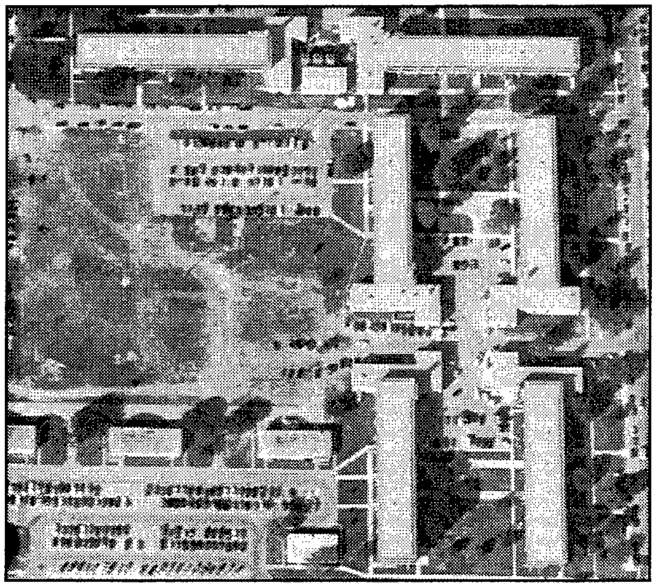


Figure B-297: T_FHOV927 image

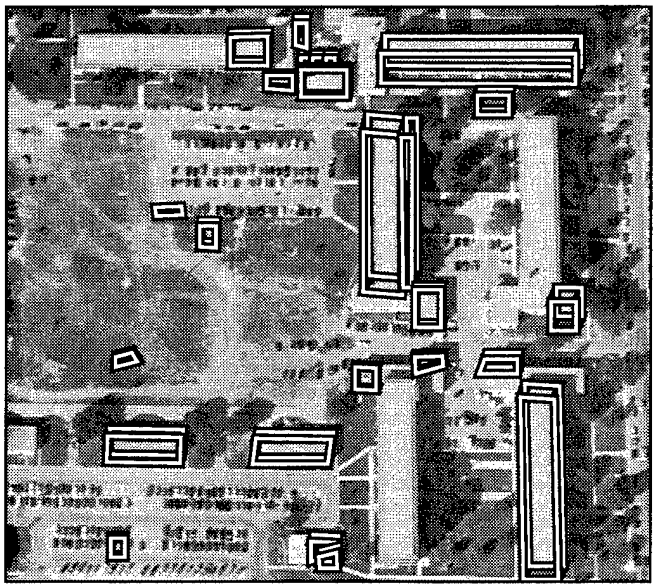


Figure B-298: BUILD+SHAVE results, T_FHOV927

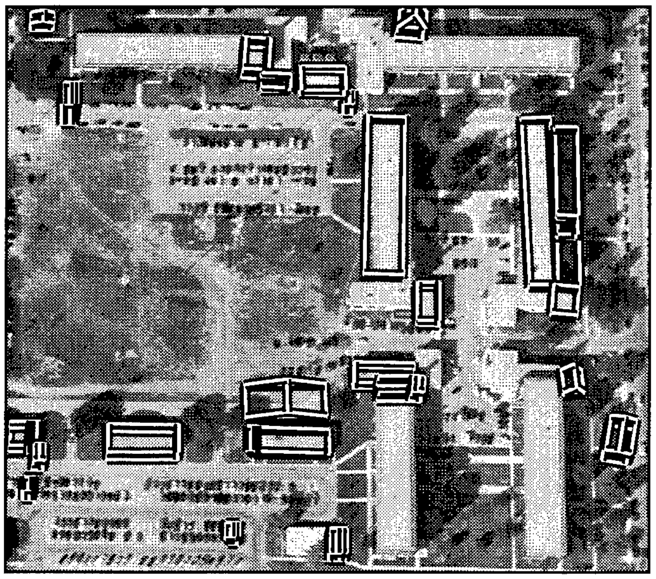


Figure B-299: VHBUILD results, T_FHOV927

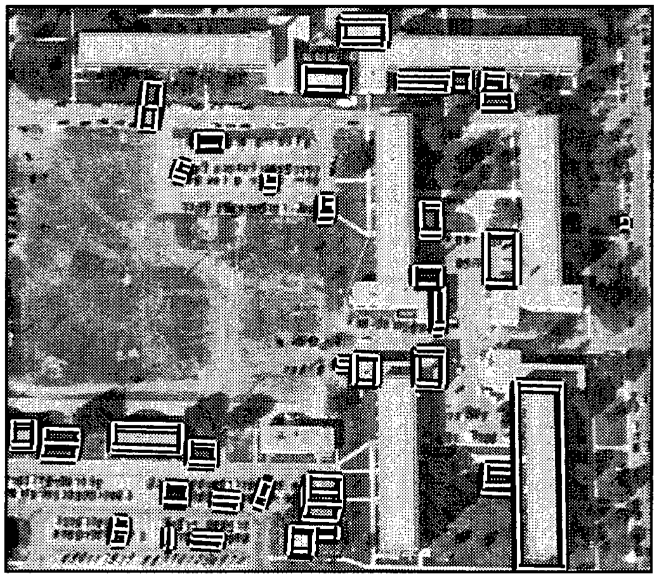


Figure B-300: PIVOT results, T_FHOV927

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	43.31	**0.148**	40.71	-	-	-
BUILD+SHAVE	**50.10**	0.294	**43.68**	**43.79**	**1.099**	**29.56**
VHBUILD	35.15	0.596	29.06	13.73	1.561	11.31
PIVOT	22.99	1.174	18.10	13.74	1.567	11.31

Table B-79: 2D and 3D evaluation results for T_FHOV927

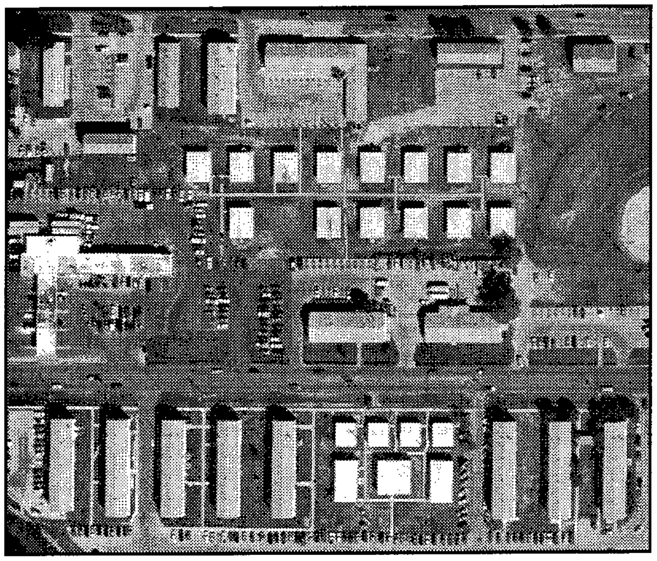


Figure B-301: VANILLA_FHN711 image

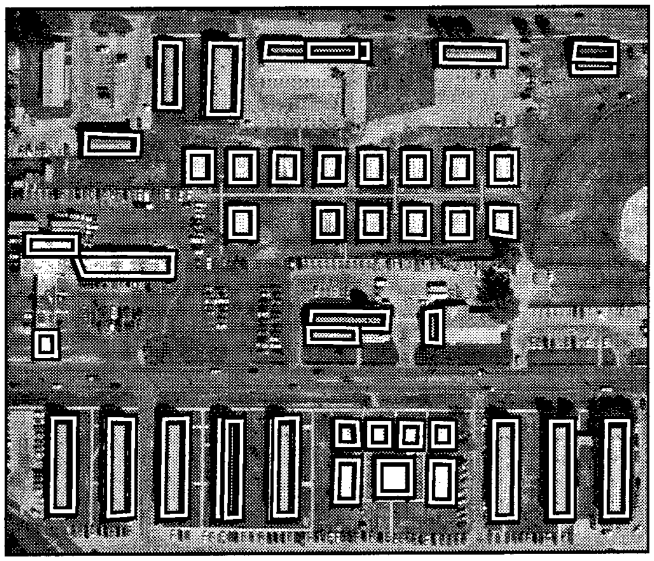


Figure B-302: BUILD+SHAVE results, VANILLA_FHN711

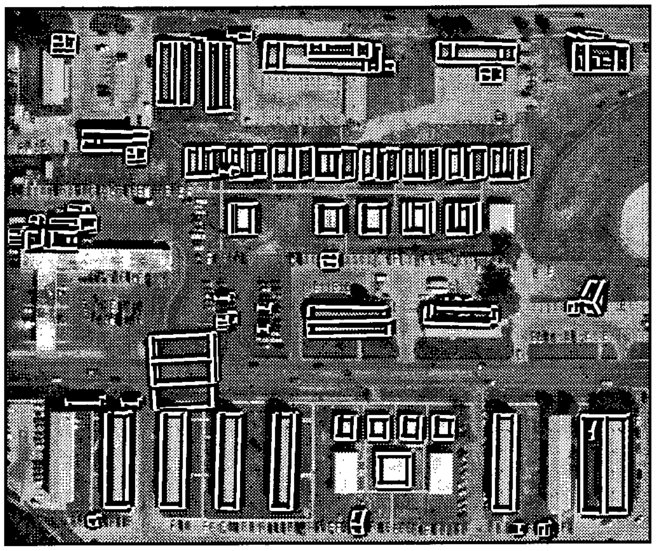


Figure B-303: VHBUILD results, VANILLA_FHN711

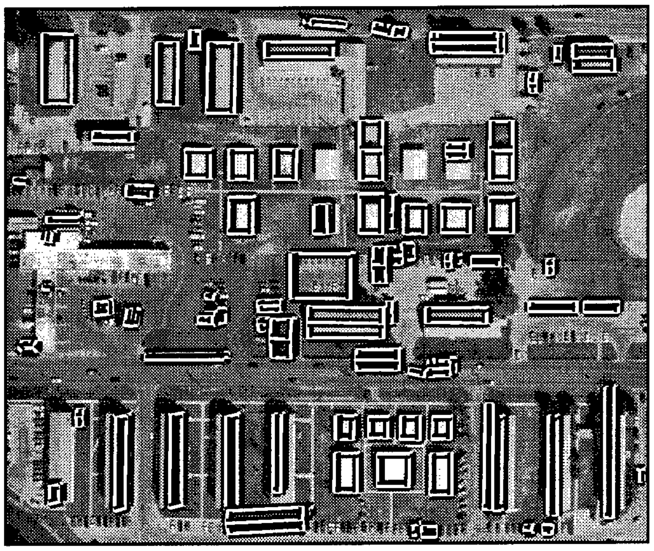


Figure B-304: PIVOT results, VANILLA_FHN711

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	81.77	**0.027**	**80.02**	-	-	-
BUILD+SHAVE	**83.87**	0.081	78.55	**73.61**	**0.326**	**59.35**
VHBUILD	70.49	0.520	51.58	51.03	5.418	13.55
PIVOT	57.91	0.680	41.55	48.39	1.500	28.04

Table B-80: 2D and 3D evaluation results for VANILLA_FHN711

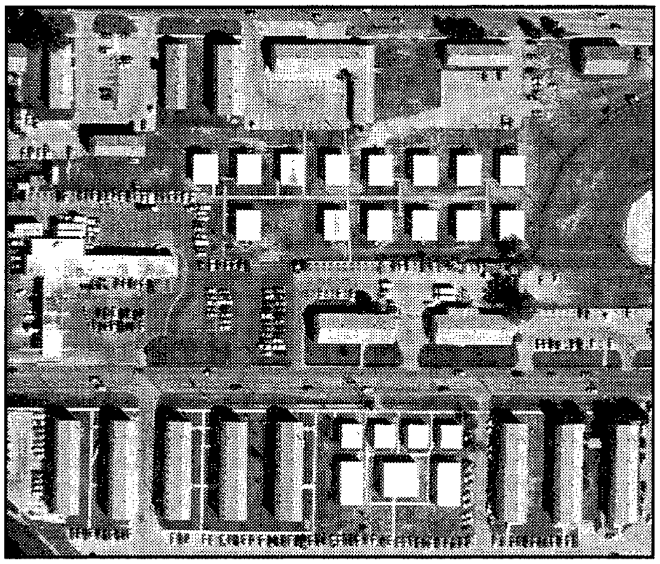


Figure B-305: VANILLA_FHN713 image

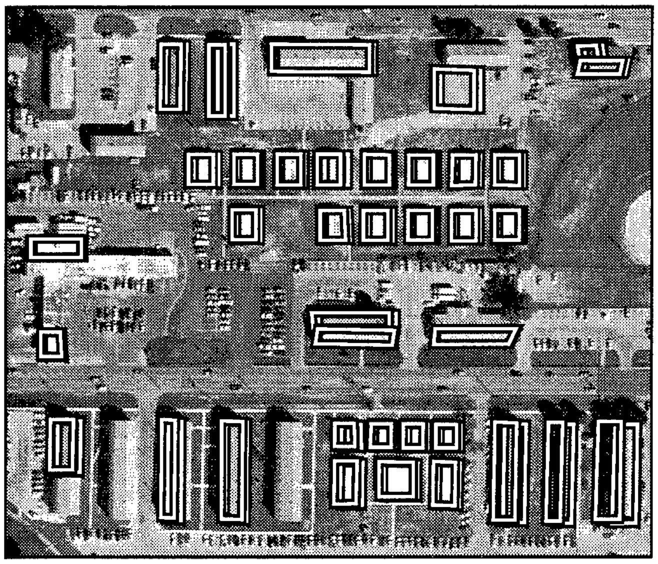


Figure B-306: BUILD+SHAVE results, VANILLA_FHN713

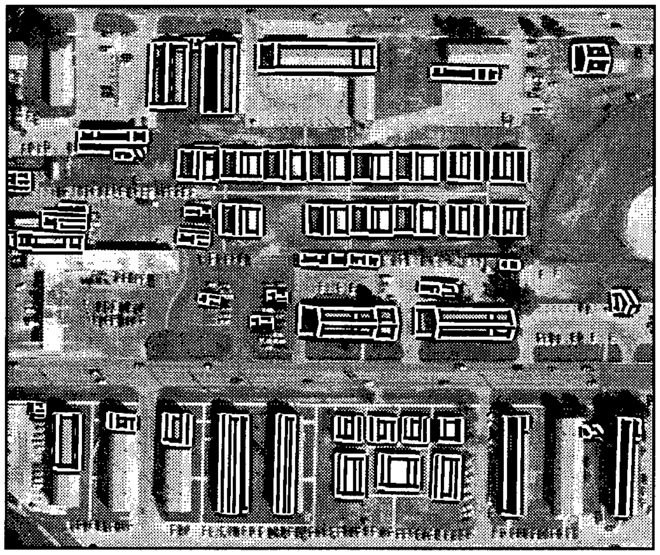


Figure B-307: VHBUILD results, VANILLA_FHN713

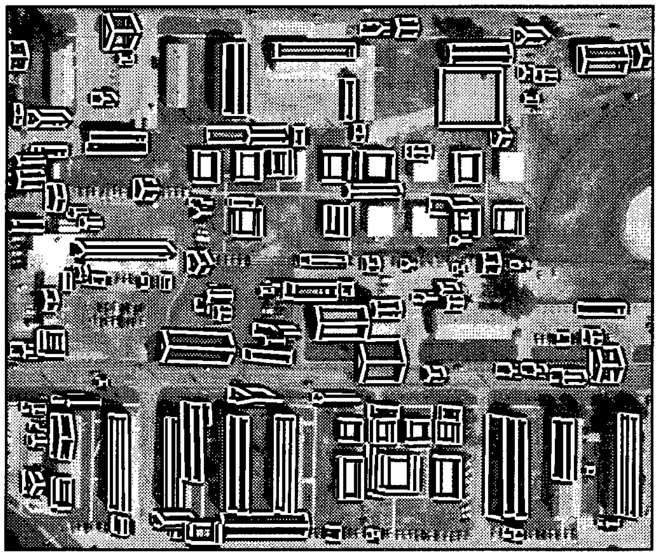


Figure B-308: PIVOT results, VANILLA_FHN713

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	59.31	**0.066**	57.07	-	-	-
BUILD+SHAVE	**66.28**	0.196	**58.65**	**49.36**	**0.831**	**35.01**
VHBUILD	64.20	0.461	49.54	40.05	3.938	15.54
PIVOT	60.30	1.186	35.15	46.45	2.706	20.58

Table B-81: 2D and 3D evaluation results for VANILLA_FHN713

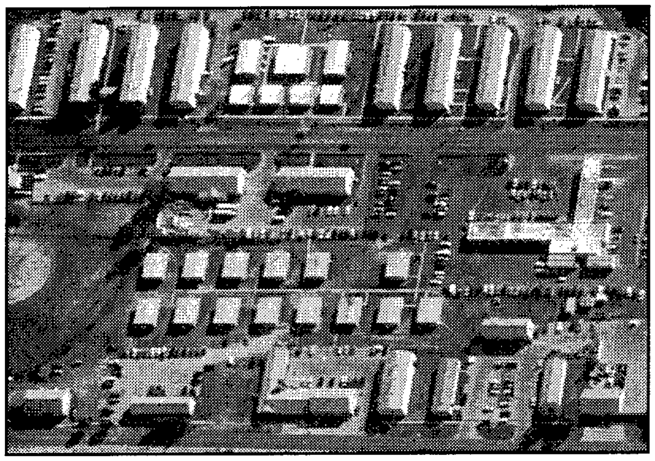


Figure B-309: VANILLA_FHOV1627 image

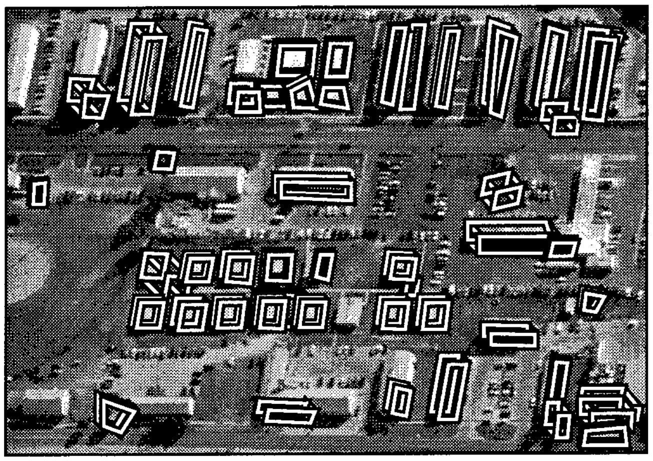


Figure B-310: BUILD+SHAVE results, VANILLA_FHOV1627

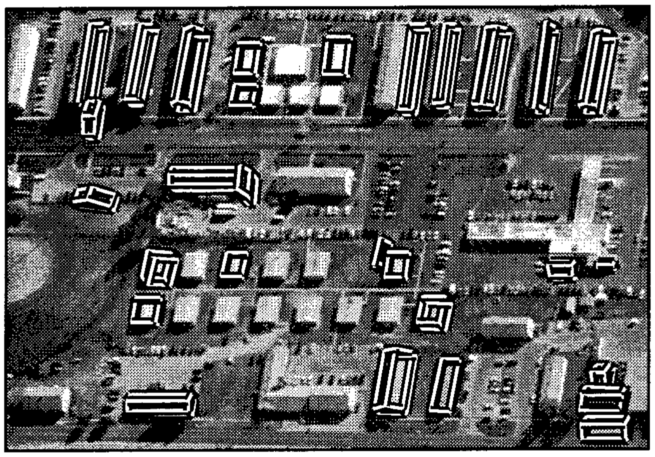


Figure B-311: VHBUILD results, VANILLA_FHOV1627

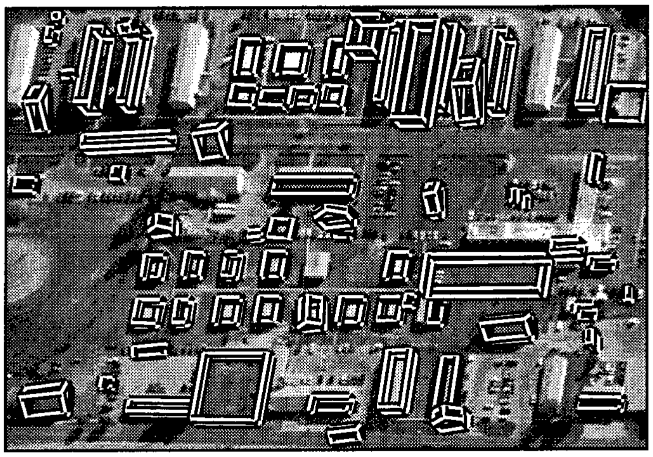


Figure B-312: PIVOT results, VANILLA_FHOV1627

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	47.39	**0.238**	42.59	-	-	-
BUILD+SHAVE	57.83	0.520	**44.45**	35.03	1.778	21.58
VHBUILD	41.83	0.337	36.67	33.36	**0.595**	**27.84**
PIVOT	**57.85**	0.955	37.26	**38.96**	2.006	21.87

Table B-82: 2D and 3D evaluation results for VANILLA_FHOV1627

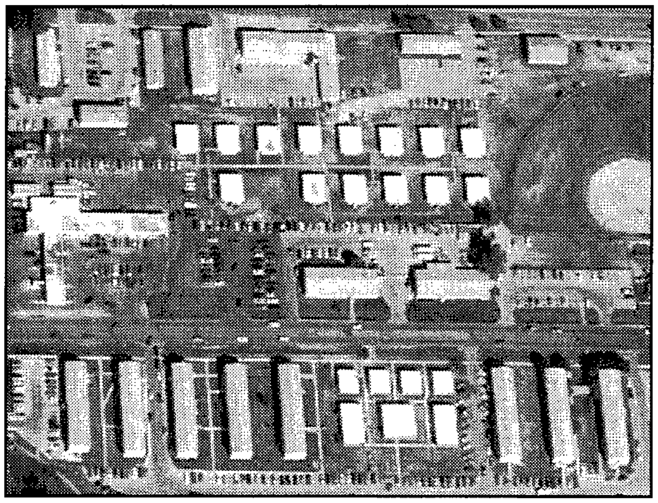


Figure B-313: VANILLA_FHOV525 image

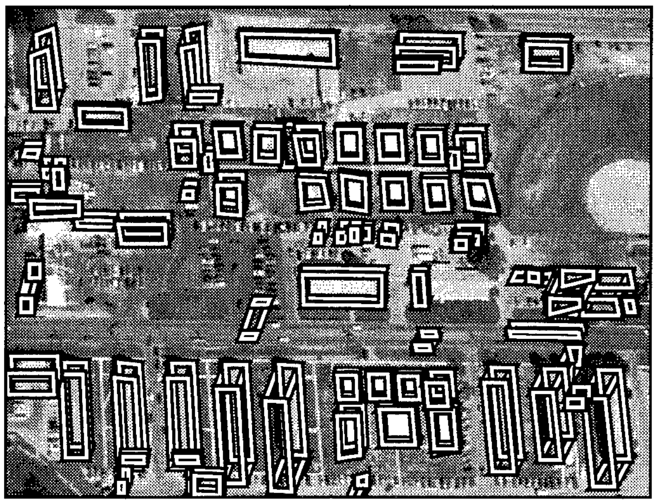


Figure B-314: BUILD+SHAVE results, VANILLA_FHOV525

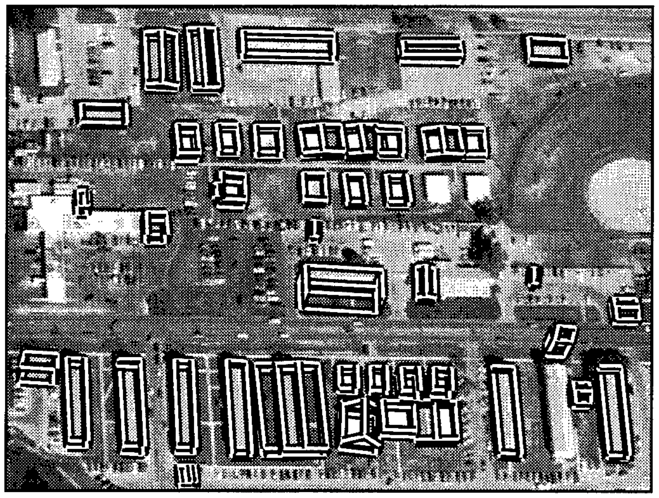


Figure B-315: VHBUILD results, VANILLA_FHOV525

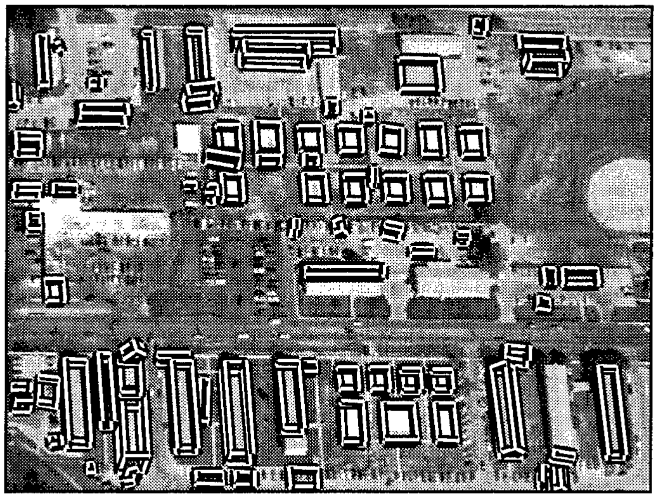


Figure B-316: PIVOT results, VANILLA_FHOV525

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	79.21	**0.220**	**67.45**	-	-	-
BUILD+SHAVE	**86.19**	0.701	53.72	55.22	4.435	16.01
VHBUILD	73.29	0.461	54.79	**58.46**	1.498	31.17
PIVOT	67.86	0.572	48.88	53.78	**1.238**	**32.28**

Table B-83: 2D and 3D evaluation results for VANILLA_FHOV525

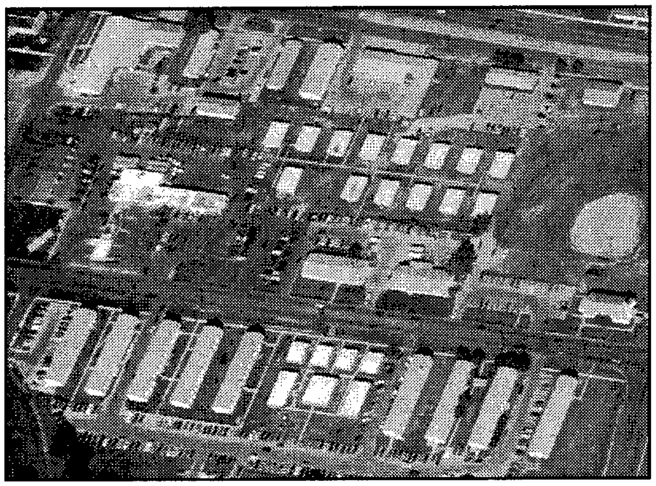


Figure B-317: VANILLA_FHOV625 image

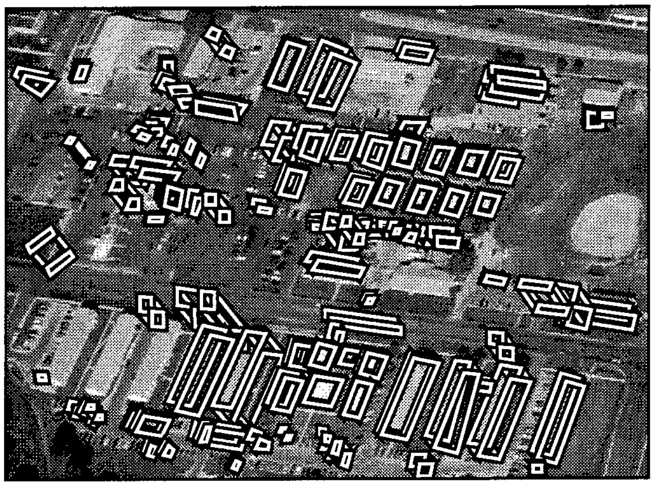


Figure B-318: BUILD+SHAVE results, VANILLA_FHOV625

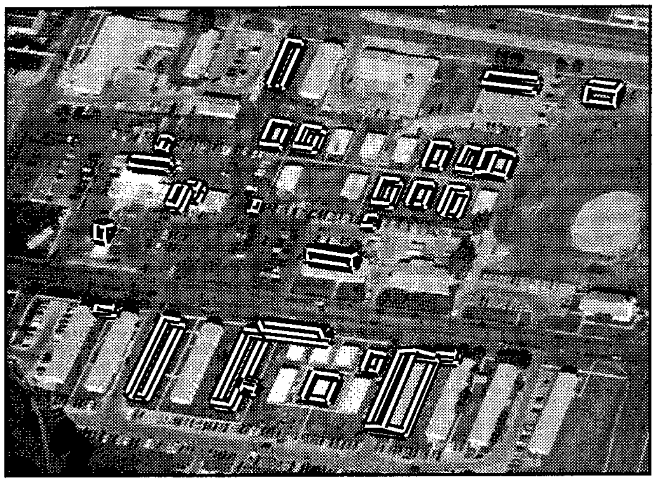


Figure B-319: VHBUILD results, VANILLA_FHOV625

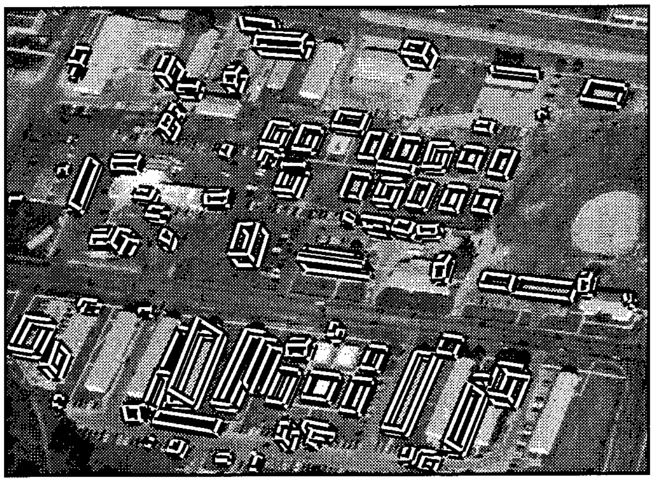


Figure B-320: PIVOT results, VANILLA_FHOV625

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	50.24	**0.448**	**41.01**	-	-	-
BUILD+SHAVE	**58.26**	1.178	34.55	**32.44**	4.838	12.63
VHBUILD	27.53	0.500	24.20	20.60	**0.931**	**17.28**
PIVOT	40.02	1.344	26.02	28.57	2.794	15.89

Table B-84: 2D and 3D evaluation results for VANILLA_FHOV625

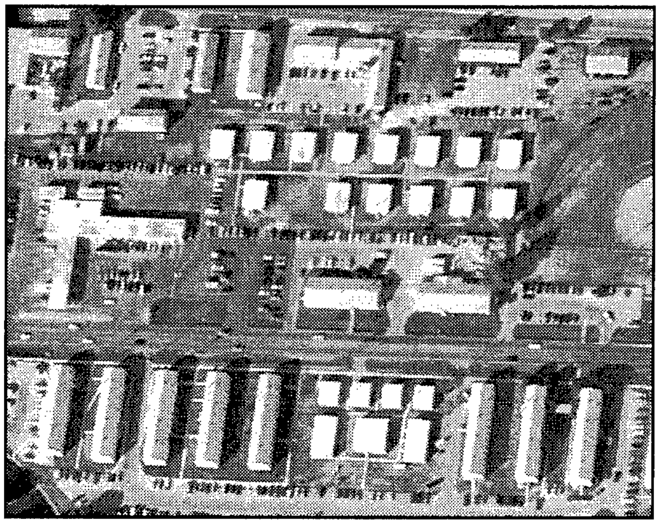


Figure B-321: VANILLA_FHOV927 image

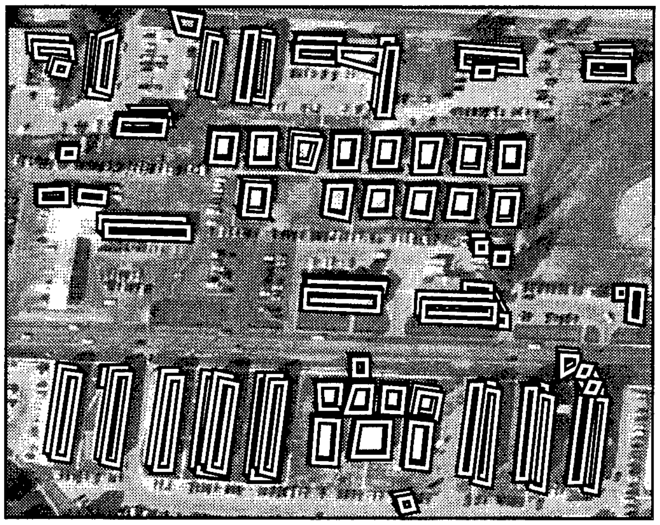


Figure B-322: BUILD+SHAVE results, VANILLA_FHOV927

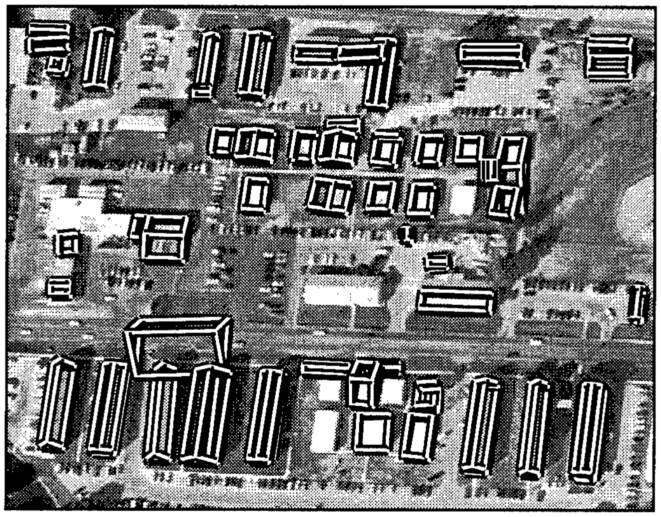


Figure B-323: VHBUILD results, VANILLA_FHOV927

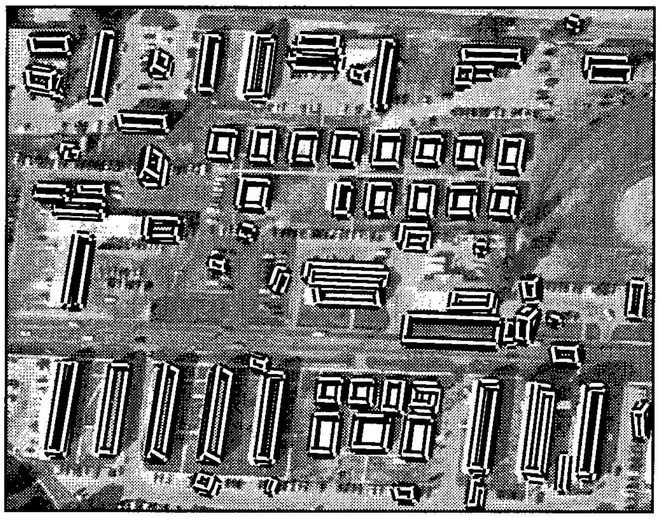


Figure B-324: PIVOT results, VANILLA_FHOV927

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	74.86	**0.120**	**68.71**	-	-	-
BUILD+SHAVE	**80.50**	0.340	63.22	56.87	1.090	35.10
VHBUILD	72.66	0.407	56.07	**60.77**	0.975	**38.16**
PIVOT	64.22	0.490	48.86	46.81	**0.943**	32.48

Table B-85: 2D and 3D evaluation results for VANILLA_FHOV927

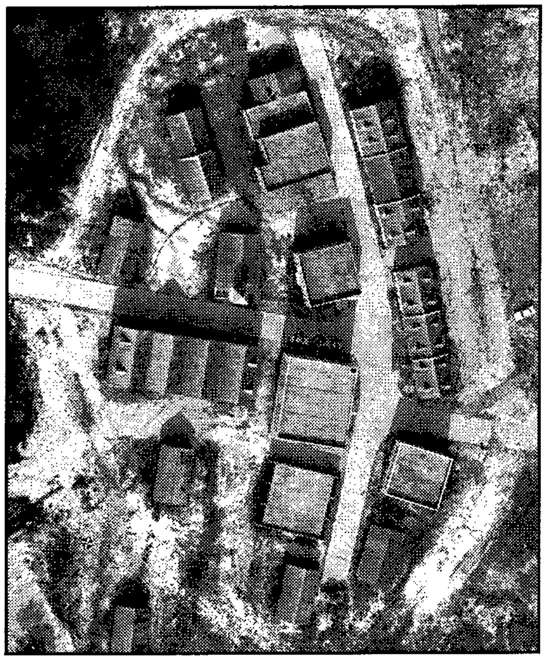


Figure B-325: VILLAGE_FTBENN407 image

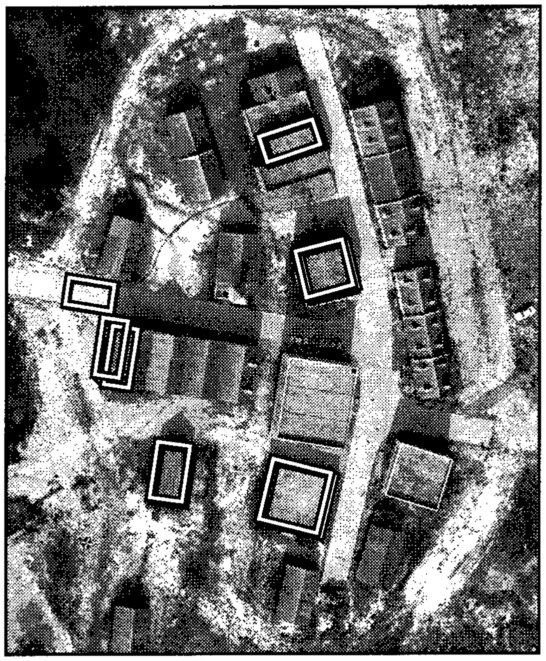


Figure B-326: BUILD+SHAVE results, VILLAGE_FTBENN407

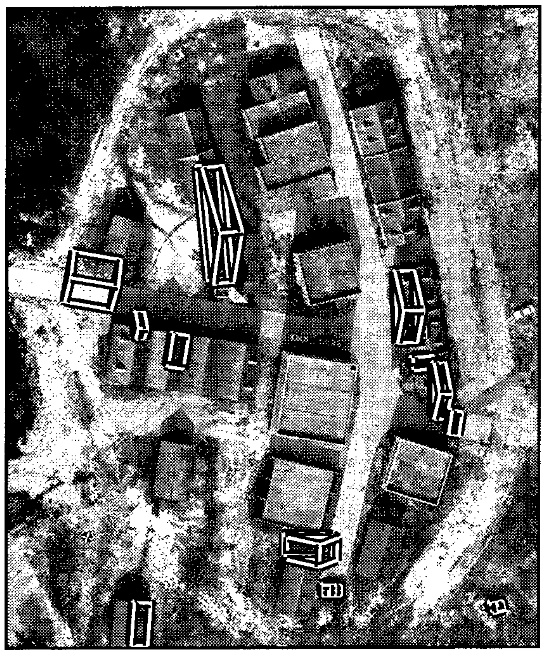


Figure B-327: VHBUILD results, VILLAGE_FTBENN407

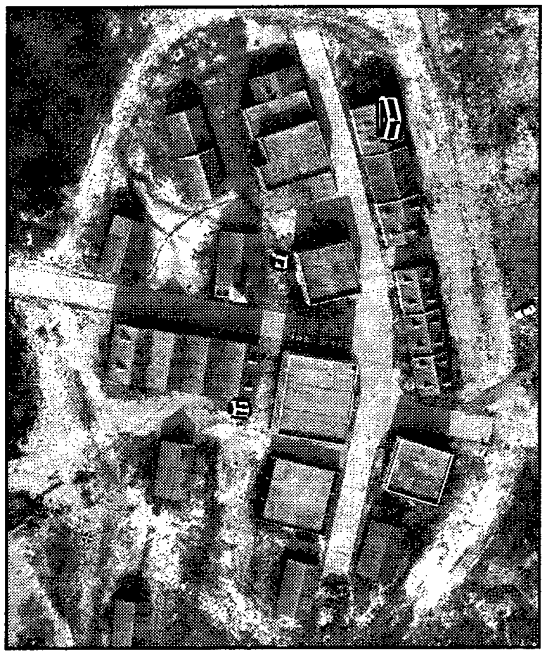


Figure B-328: PIVOT results, VILLAGE_FTBENN407

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	17.36	**0.130**	16.97	-	-	-
BUILD+SHAVE	**18.52**	0.234	**17.75**	**12.26**	1.156	**10.74**
VHBUILD	10.62	1.143	9.47	5.61	4.532	4.48
PIVOT	1.06	0.545	1.05	0.71	**1.124**	0.70

Table B-86: 2D and 3D evaluation results for VILLAGE_FTBENN407

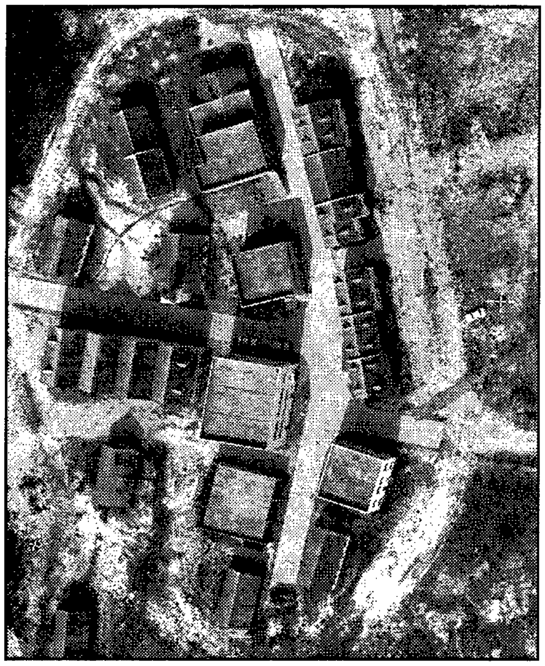


Figure B-329: VILLAGE_FTBENN408 image

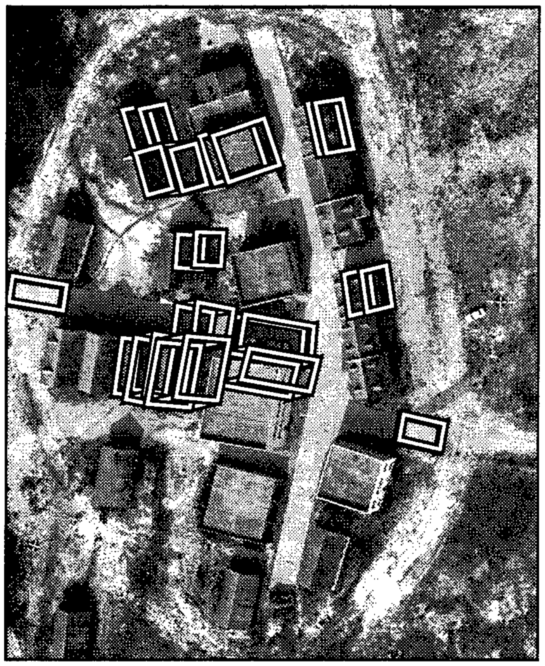


Figure B-330: BUILD+SHAVE results, VILLAGE_FTBENN408

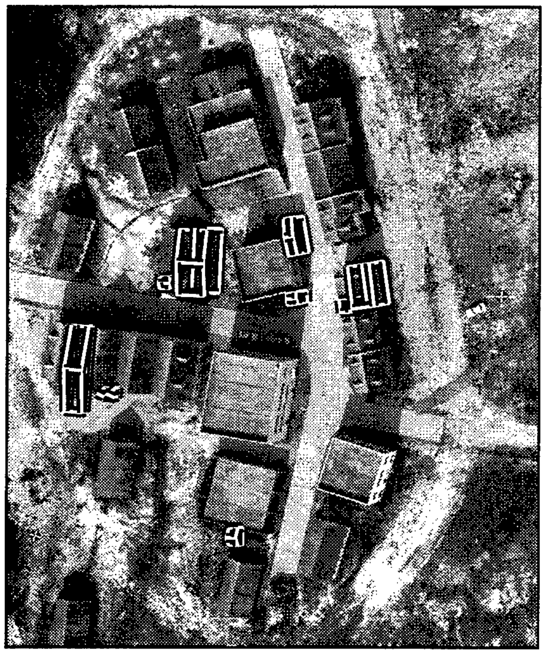


Figure B-331: VHBUILD results, VILLAGE_FTBENN408

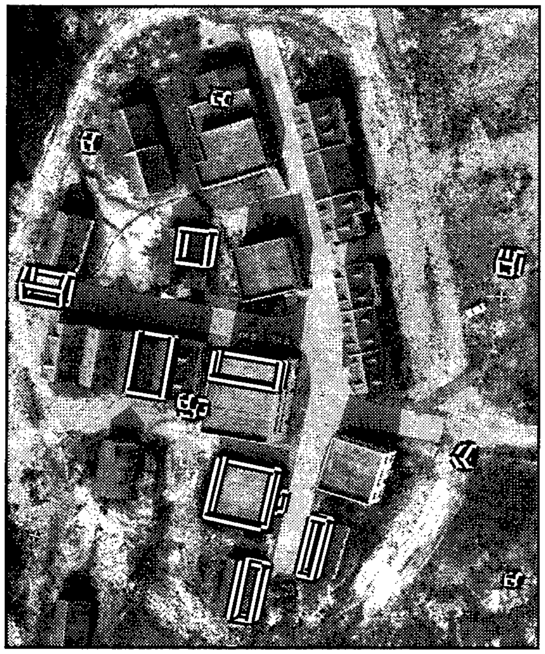


Figure B-332: PIVOT results, VILLAGE_FTBENN408

system	2D bld	2D brf	2D qual	3D bld	3D brf	3D qual
BUILD	22.85	0.422	20.84	-	-	-
BUILD+SHAVE	**27.03**	0.701	**22.72**	**19.93**	2.297	**13.67**
VHBUILD	7.86	0.606	7.50	3.32	2.336	3.08
PIVOT	19.17	**0.320**	18.06	8.58	**0.798**	8.03

Table B-87: 2D and 3D evaluation results for VILLAGE_FTBENN408